Lecture Notes in Artificial Intelligence 5402

Edited by R. Goebel, J. Siekmann, and W. Wahlster

Subseries of Lecture Notes in Computer Science

Gérard Huet Amba Kulkarni
Peter Scharf (Eds.)

Sanskrit Computational Linguistics

First and Second International Symposia
Rocquencourt, France, October 29-31, 2007
Providence, RI, USA, May 15-17, 2008
Revised Selected and Invited Papers



Springer

Series Editors

Randy Goebel, University of Alberta, Edmonton, Canada
Jörg Siekmann, University of Saarland, Saarbrücken, Germany
Wolfgang Wahlster, DFKI and University of Saarland, Saarbrücken, Germany

Volume Editors

Gérard Huet
INRIA, Centre de Recherche Paris–Rocquencourt
Domaine de Voluceau, BP 105, 78153 Le Chesnay CEDEX, France
E-mail: gerard.huet@inria.fr

Amba Kulkarni
University of Hyderabad
Department of Sanskrit Studies
Hyderabad 500046, India
E-mail: apksh@uohyd.ernet.in

Peter Scharf
Brown University
Department of Classics
PO Box 1856, Providence, RI 02912, USA
E-mail: scharf@brown.edu

Library of Congress Control Number: 2008943854

CR Subject Classification (1998): J.5, H.3.1, I.2.7, F.4.2

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN 0302-9743
ISBN-10 3-642-00154-8 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-00154-3 Springer Berlin Heidelberg New York

springer.com

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12614622 06/3180 5 4 3 2 1 0

Foreword

Sanskrit is the primary culture-bearing language of India, with a continuous production of literature in all fields of human endeavor over the course of four millennia. Preceded by a strong oral tradition of knowledge transmission, records of written Sanskrit remain in the form of inscriptions dating back to the first century B.C.E. Extant manuscripts in Sanskrit number over 30 million, one hundred times those in Greek and Latin combined, constituting the largest cultural heritage that any civilization has produced prior to the invention of the printing press. Sanskrit works include extensive epics; subtle and intricate philosophical, mathematical, medical, legal, and scientific treatises; and imaginative and rich literary, poetic, and dramatic texts. While the Sanskrit language is of preeminent importance to the intellectual and cultural heritage of India, the importance of the intellectual and cultural heritage of India to the rest of the world during the past few millennia and in the present era can hardly be overestimated. The intellectual and cultural heritage of India has been a major factor in the development of the world's religions, languages, literature, arts, sciences, and history.

Sanskrit documents are moving into the digital medium. Recent decades have witnessed the growth of machine-readable Sanskrit texts in archives such as the Thesaurus Indogermanischer Text- und Sprachmaterialien (TITUS),[1] Kyoto University,[2] Indology,[3] the Göttingen Register of Electronic Texts in Indian Languages.[4] The last few years have witnessed a burgeoning of digital images of Sanskrit manuscripts and books hosted on-line. For example, the University of Pennsylvania Library, which houses the largest collection of Sanskrit manuscripts in the Western Hemisphere, has made digital images of 33 of them available electronically,[5] The Universal Digital Library[6] and Google Books[7] have made digital images of large numbers of Sanskrit texts accessible as part of their enormous library digitization projects. Digitized Sanskrit documents include machine-readable text and images of lexical resources such as those of the Cologne Digital Sanskrit Lexicon project[8] and the University of Chicago's Digital Dictionaries of South Asia project.[9]

In the first half of the first millenium B.C.E., there developed a tradition of linguistic analysis in India that included metrics (*chandas*), etymology (*nirukta*),

---

[1] http://titus.uni-frankfurt.de/
[2] ftp://ccftp.kyoto-su.ac.jp/pub/doc/sanskrit/
[3] http://indology.info/
[4] http://www.sub.uni-goettingen.de/ebene_1/fiindolo/gretil.htm
[5] http://oldsite.library.upenn.edu/etext/sasia/skt-mss/
[6] http://www.ulib.org/
[7] http://books.google.com/
[8] http://www.sanskrit-lexicon.uni-koeln.de/
[9] http://dsal.uchicago.edu/dictionaries/list.html#sanskrit

phonetics (*śikṣā*), and grammar (*vyākaraṇa*). By the early fourth century B.C.E., Pāṇini, who was born in Śālātura, near modern Lahore, had completed the *Aṣṭādhāyī* (literally, the book in eight chapters), which remains to this day the most authoritative treatise on Sanskrit grammar. It consists of nearly 4,000 rules that give a precise and fairly complete description of late Vedic Sanskrit.

Pāṇini's grammar was complemented by Patañjali's commentary *Mahābhāṣya* (2nd century B.C.E.), incorporating an earlier commentary of Kātyāyana (3rd century B.C.E.). The work of these three authors (referred to collectively as the *trimuni* 'wise triad') became de facto a prescriptive grammar for what became Classical Sanskrit, and thus the gold standard for Sanskrit grammaticality. It is therefore natural that scientists interested in the formal modelling of Sanskrit refer to Pāṇinian authority, and discuss ways of formally capturing Pāṇinian methods.

As oral, manuscript, and print media that have conveyed the knowledge embodied in the ancient Sanskrit language make their transition into digital media, the modern discipline of computational linguistics opens a new domain of investigation: Sanskrit computational linguistics. Scholars in a variety of disciplines are finding interest in this field for various reasons. Linguists, for example, are finding new challenges in formalizing the syntax of a free-word-order language, computer scientists are drawn to model techniques of generative grammar used by the ancient India grammarian Pāṇini, philologists are using digital methods to assist in critical editing, and collaboration among scholars in these various disciplines is fostering new ideas and building corpora, databases, and tools for the use of academic researchers and commercial enterprises. These developments have encouraged the editors of the present volume themselves to develop an integrated digital Sanskrit Library[10] and software for processing Sanskrit.[11]

The discovery by the editors of each other's work led to their mutual collaboration and to a decision to encourage all scholars interested in digital work with Sanskrit text–whether they be linguists, philologists, computer scientists, or indologists–to share their work at regular Sanskrit Computational Linguistics symposia. The present volume is a collection of papers delivered at the first two such symposia: the First International Sanskrit Computational Linguistics Symposium hosted by Gérard Huet and Amba Kulkarni October 29–31, 2007 at the Paris-Rocquencourt Center of INRIA (Institut National de Recherche en Informatique et en Automatique, France), and the Second International Sanskrit Computational Linguistics Symposium hosted by Peter Scharf May 15–17, 2008 at Brown University, Providence, RI, USA.[12]

[10] http://sanskritlibrary.org

[11] http://sanskrit.inria.fr/, http://sanskrit.uohyd.ernet.in/

[12] The Second International Sanskrit Computational Linguistics Symposium was arranged as part of the International Digital Sanskrit Library Integration project at Brown funded by the U.S. National Science Foundation, Division of Information and Intelligent Systems, under grant number 0535207. Any opinions, findings, and conclusions or recommendations expressed are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Papers in the volume touch on several topics: Pāṇinian studies, computational linguistics, lexicography, philology, and OCR. The first three papers investigate the overall structure of the Pāṇinian grammatical system. George Cardona from the University of Pennsylvania, the keynote speaker at the second symposium, in his paper "On the Structure of Pāṇini's System,"‡ demonstrates that Pāṇini derives utterances, rather than individual words, in a continuum that integrates syntax inextricably with inflectional and derivational morphology. The second paper, "On the Architecture of Pāṇini's System,"† is authored by Paul Kiparsky from Stanford University. He defends, in contrast, a description of Pāṇinian devices layered in four levels of organization beginning with semantics and ending with phonology, interconnected by three interfaces. He argues that Pāṇini was able to capture essential generalizations about the language from his requirement that the description be as simple as possible. In "Modeling Pāṇinian Grammar,"† Peter Scharf from Brown University examines how to model various features of Pāṇinian grammar. He argues that semantics comprise a rich portion of the grammar, and that Pāṇini essentially conceives of no more than two levels: meaning and sound.

The next two papers aim at presenting in detail computational processes designed to simulate the derivation of forms from Pāṇini's sūtras. First, Anand Mishra from Heidelberg University, in his paper "Simulating the Pāṇinian System of Sanskrit Grammar"†, emulates the operation of rules in the *Aṣṭādhāyī* in the model of a lexical database. Second, Pawan Goyal from IIT Kanpur, Amba Kulkarni from the University of Hyderabad, and Laxmidhar Behera from IIT Kanpur and the University of Ulster present a method to implement rule selection procedures in their paper "Computer Simulation of Aṣṭādhāyī: Some Insights"‡. They propose to use modern programming features, such as event-driven programming, and to organize memory in data spaces, to simulate conditions of rule application and resolve conflicts.

A number of papers report on computer implementation of Sanskrit analysis systems. Gérard Huet from the Paris-Rocquencourt INRIA Center, in his paper "Formal Structure of Sanskrit Text: Requirements Analysis for a Mechanical Sanskrit Processor,"‡ provides a general overview of work in formal linguistics and computer science relevant to design issues in Sanskrit computational linguistics and proposes a roadmap for the development of inter-operable modules. Pawan Goyal, Vipul Arora, and Laxmidhar Behera from IIT Kanpur, in their paper "Analysis of Sanskrit Text: Parsing and Semantic Relations,"† describe their implementation of a dependency parser for Sanskrit that relies on finite-state techniques for morpho-phonetic analysis. The paper "Inflexional Morphology Analyser for Sanskrit,"† by Girish Nath Jha from Jawaharlal Nehru University in Delhi, together with his students Muktanand Agrawal, Subash, Sudhir K. Mishra, Diwakar Mani, Diwakar Mishra, Manji Bhadra, and Surjit K. Singh, describes the morphology analyzer module from an ongoing effort at JNU to develop a comprehensive Sanskrit processing platform. The next paper, "Semantic Processing in Pāṇini's Kāraka System"‡ by Girish Nath Jha from JNU and Sudhir K. Mishra from CDAC Pune, describes the component of this

platform that deals with the interface between syntax and semantics, in particular with Pāṇini's kāraka rules. Then Malcolm Hyman, from the Max Planck Institute for the History of Science in Berlin, in his paper "From Pāṇinian Sandhi to Finite State Calculus,"[†] describes the use of finite state cascading transducers to produce an efficient implemention of external sandhi consistent with Pāṇini's description. Finally, in his paper "SanskritTagger: A Stochastic Lexical and POS Tagger for Sanskrit,"[†] Oliver Hellwig, from the Freie Universität in Berlin, describes the use of statistical techniques to implement a Sanskrit part of speech tagger whose parameters have been trained over a large manually annotated corpus. These six papers span a large spectrum of computer science concepts and techniques to achieve actual processing of realistic Sanskrit sentences. They indeed represent well the state of the art of effective Sanskrit computational linguistic processors.

The next group of papers deal with specific issues in the formal description of Sanskrit grammar. Prasad Joshi from Fergusson College in Pune, in his paper "A Glimpse into the *Apadam*-Constraint in the Tradition of Sanskrit Grammar,"[‡] argues that the Pāṇinian principle that words end in inflectional terminations serves as a sentential grammaticality constraint. Pawan Goyal and R. Mahesh K. Sinha from IIT Kanpur present their adaptation of the AnglaBharati machine translation system to Sanskrit in their paper, "A Study Towards Design of an English to Sanskrit Machine Translation System."[‡] One of the problems of implementing Pāṇini's sūtras as a generative device is the generation of undesirable forms by the unrestricted application of rules. Malhar Kulkarni, from IIT Mumbai, examines the particular case of sandhi rules that produce nasalization and phonetic doubling in his paper "Phonological Overgeneration in Pāṇinian System."[†] Boris Oguibénine from Université de Strasbourg presents in his paper "Issues in Combinatorial Analysis of Vedic Verbal and Nominal Forms"[‡] the problem of lemmatizing Vedic inflected forms, and argues that analyzing word-final phonetic strings right to left reveals distinctions more efficiently, from a computational perspective, than typical morphemic analysis.

We then offer two contributions to the construction of lexical resources for Sanskrit. First Malhar Kulkarni and Pushpak Bhattacharya from IIT Mumbai, in their paper "Verbal Roots in the Sanskrit Wordnet,"[‡] describe their progress in developing a Wordnet-like Sanskrit lexical database based on verbal roots and their classification in dhātupāṭhas. Then S. Varakhedi from the Sanskrit Academy in Hyderabad, V. Jaddipal from the Rāṣṭriya Sanskrit Vidyāpīṭha in Tirupati, and V. Sheeba from the University of Hyderabad, in their paper "An Effort to Develop a Tagged Lexical Resource for Sanskrit,"[†] describe the problems encountered in their digitalization of the monumental Sanskrit encyclopedia *Vācaspatyam* by Pandit Tārānātha Tarkavācaspati.

Five papers deal with issues crucial to the integration of digital images with machine-readable texts. Three of these concern the use of computational procedures in critical editing, and two deal with optical character recognition (OCR) for Indic scripts. Peter Robinson from the University of Birmingham, in

his paper, "Towards a Scholarly Collation System for the Next Decades,"[‡] explains desiderata for computer-based manuscript collation and argues that the digital revolution is transforming scholarly editing into a collaborative enterprise. He reviews the history of his Collate software, and his participation in the development of more comprehensive philology tools. Then Marc Csernel and François Patte from Université Paris Descartes tell of their experience in designing a distance between versions of a Sanskrit text in their paper "Critical Edition of Sanskrit Texts."[†] Finally, Wendy Phillips-Rodriguez from the National Autonomous University of Mexico and Christopher Howe and Heather Windram from the University of Cambridge investigate statistical techniques and phylogenetic algorithms for use in Sanskrit philology in their paper "Chi-squares and the Phenomenon of 'Change of Exemplar' in the *Dyūtaparvan*."[‡]

Two papers in this collection are concerned with the production of machine-readable text from digital images of Sanskrit using OCR technology. First, Thomas Breuel from the Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI) and the University of Kaiserslautern, in his paper "Applying the OCRopus OCR System to Scholarly Sanskrit Literature,"[‡] describes the use of weighted finite state transducers to produce statistical optimization in the OCRopus open-source multi-lingual modular OCR system. Second, Anurag Bhardwaj, Srirangaraj Setlur, and Venu Govindaraju from the University of Buffalo describe their use of keyword spotting, as an alternative to OCR, to retrieve information in documents that mix languages and scripts in their paper "Keyword Spotting Techniques for Sanskrit Documents."[‡]

This collection ends with a contribution by R. K. Joshi from the Centre for Development of Advanced Computing (C-DAC) in Mumbai, T. N. Dharmadhikari from the Vaidika Sanshodan Mandal in Pune, and Vijay Vasudev Bedekar from the Institute of Oriental Study in Thane, entitled "The Phonemic Approach for Sanskrit Text."[†] Professor Joshi presented his proposal for a phonemic encoding of Sanskrit as his contribution to the workshop at the First International Sanskrit Computational Linguistics Symposium in October 2007 in Rocquencourt. This presentation was one of his last professional appearances; sadly he passed away a few months later, en route to a Unicode Technical Committee meeting in California. As a calligrapher, a designer, a poet, a researcher, and a teacher, Professor Joshi (1936–2008) was a highly respected and greatly admired scholar.

The editors would like to thank all those who have contributed to the present volume including the authors of the papers, members of the Program Committees who commented on initial drafts of papers presented at the two symposia in which the papers were first presented, and participants in the symposia whose discussion contributed to paper revisions. The papers presented at the First International Sanskrit Computational Linguistics Symposium convened by Gérard Huet and Amba Kulkarni in Rocquencourt October 29–31, 2007 are marked with a dagger: †. The papers presented at the Second International Sanskrit Computational Linguistics Symposium convened by Peter Scharf in Providence May 15–17, 2008 are marked with a double dagger: ‡. Responsibility for editing the

papers presented at each symposium fell primarily to the convenors of the respective symposia.

December 2008

Gérard Huet
Amba Kulkarni
Peter Scharf

Organization

The first symposium was organized by Centre INRIA Paris-Rocquencourt in joint collaboration with the Department of Sanskrit Studies, University of Hyderabad in October 2007 at INRIA, Rocquencourt.

Program Chair

Gérard Huet	INRIA, Rocquencourt, France
Amba Kulkarni	University of Hyderabad, Hyderabad, India

Program Committee

Pushpak Bhattacharya	Computer Science and Engineering Department, IIT Mumbai, India
Brendan S. Gillon	Department of Linguistics, McGill University, Montreal, Canada
Jan Houben	École Pratique des Hautes Études, Paris, France
Gérard Huet	Centre INRIA Paris-Rocquencourt, France (Co-chair)
Girish Nath Jha	Special Centre for Sanskrit Studies, J.N.U. New Delhi, India
Amba Kulkarni	Department of Sanskrit Studies, University of Hyderabad, India (Co-chair)
Malhar Kulkarni	Dept. of Humanities & Social Sciences, IIT Mumbai, India
Alain Lecomte	UFR Sciences du langage, Université Paris 8, Saint-Denis, France
Narayana Murthy Kavi	Dept. of Computer and Information Sciences, University of Hyderabad, India
Georges-Jean Pinault	École Pratique des Hautes Études, Paris, France
K.V.R. Krishnamacharyulu	Sanskrit University, Jaipur, India
Peter M. Scharf	Department of Classics, Brown University, Providence, USA
Shivamurthy Swamiji	Sri Taralabalu Jagadguru Brihanmath, Sirigere, India
Muneo Tokunaga	Graduate School of Letters, Kyoto University, Kyoto, Japan
Lalit Kumar Tripathi	Rashtriya Sanskrit Sansthan, Allahabad, India
Srinivasa Varakhedi	NLP Department, Rashtriya Sanskrit Vidyapeetha, Tirupati, India

The second symposium was organized by the Department of Classics, Brown University, in May 2008.

Program Chair

Peter M. Scharf	Department of Classics, Brown University, Providence, USA

Program Committee

Brendan S. Gillon	Department of Linguistics, McGill University, Montreal, Canada
Venugopal Govindaraju	Center for Excellence in Document Analysis and Recognition, University of Buffalo, USA
Malcolm Hyman	Max-Planck-Institut für Wissenschaftsgeschichte, Berlin, Germany
Gérard Huet	Centre INRIA Paris-Rocquencourt, France
Amba Kulkarni	Department of Sanskrit Studies, University of Hyderabad, India
Malhar Kulkarni	Dept. of Humanities & Social Sciences, IIT Mumbai, India
Elli Mylonas	Computing and Information Services, Brown University, Providence, USA

Table of Contents

On the Structure of Pāṇini's System 1
George Cardona

On the Architecture of Pāṇini's Grammar 33
Paul Kiparsky

Modeling Pāṇinian Grammar 95
Peter M. Scharf

Simulating the Pāṇinian System of Sanskrit Grammar 127
Anand Mishra

Computer Simulation of *Aṣṭādhyāyī*: Some Insights 139
Pawan Goyal, Amba Kulkarni, and Laxmidhar Behera

Formal Structure of Sanskrit Text: Requirements Analysis for a Mechanical Sanskrit Processor 162
Gérard Huet

Analysis of Sanskrit Text: Parsing and Semantic Relations 200
Pawan Goyal, Vipul Arora, and Laxmidhar Behera

Inflectional Morphology Analyzer for Sanskrit 219
Girish Nath Jha, Muktanand Agrawal, Subash, Sudhir K. Mishra, Diwakar Mani, Diwakar Mishra, Manji Bhadra, and Surjit K. Singh

Semantic Processing in Pāṇini's Kāraka System 239
Girish Nath Jha and Sudhir K. Mishra

From Pāṇinian Sandhi to Finite State Calculus 253
Malcolm D. Hyman

`SanskritTagger`: A Stochastic Lexical and POS Tagger for Sanskrit 266
Oliver Hellwig

A Glimpse into the *Apadam*-Constraint in the Tradition of Sanskrit Grammar 278
Prasad P. Joshi

A Study towards Design of an English to Sanskrit Machine Translation System 287
Pawan Goyal and R. Mahesh K. Sinha

Phonological Overgeneration in Paninian System 306
Malhar Kulkarni

Issues in Combinatorial Analysis of Vedic Verbal and Nominal Forms . . . 320
Boris Oguibénine

Verbal Roots in the Sanskrit Wordnet . . . 328
Malhar Kulkarni and Pushpak Bhattacharyya

An Effort to Develop a Tagged Lexical Resource for Sanskrit . . . 339
S. Varakhedi, V. Jaddipal, and V. Sheeba

Towards a Scholarly Editing System for the Next Decades . . . 346
Peter Robinson

Critical Edition of Sanskrit Texts . . . 358
Marc Csernel and François Patte

Chi-Squares and the Phenomenon of "Change of Exemplar" in the *Dyūtaparvan* . . . 380
Wendy J. Phillips-Rodriguez, Christopher J. Howe, and Heather F. Windram

Applying the OCRopus OCR System to Scholarly Sanskrit Literature . . . 391
Thomas M. Breuel

Keyword Spotting Techniques for Sanskrit Documents . . . 403
Anurag Bhardwaj, Srirangaraj Setlur, and Venu Govindaraju

The Phonemic Approach for Sanskrit Text . . . 417
R.K. Joshi, T.N. Dharmadhikari, and Vijay Vasudev Bedekar

Author Index . . . 425

On the Structure of Pāṇini's System

George Cardona

Department of Linguistics, University of Pennsylvania
619 Williams Hall, Philadelphia, PA 19104-6305
cardona@sas.upenn.edu

Abstract. Pāṇini accounts for utterances through a derivational procedure that starts from meaning conditions involving actions, participants in actions, and other things that are related to each other. His derivational system thereby serves to form utterances of which words are a part, not isolated words that are then strung together to form utterances. Pāṇini's system is a continuum that starts from meaning and cooccurrence conditions that determine the introduction of affixes to bases in order to form initial strings and subsequently applies additional affixation and replacement rules to produce final strings. The system does not isolate morphology as a distinct component absolutely independent of syntax. Declensional and conjugational morphology are part of syntactic derivation, and derivational morphology is generally incorporated in the syntactic machinery. Primary derivational affixes are introduced in the course of syntactic derivation. In addition, secondary derivation affixes are regularly introduced after padas—terms that contain endings introduced in syntactic derivation—and not after mere bases in a separate morphological component. Further, composition takes place within the context of syntactic derivation; compounds are formed from related padas of initial strings, which involve number distinctions. Even the formation of certain items with feminine suffixes takes place within the syntactic machinery and not in a totally separate lexicon. Thus Pāṇini's derivational system is an integrated system accounting for utterances of Sanskrit; it lacks a sharp dichotomy between what western grammarians call syntax and morphology.

Keywords: Grammatical theory, syntax, morphology, organization of grammar, Indian grammatical theory, Pāṇini, Kātyāyana, Patañjali, Bhartṛhari.

G. Huet, A. Kulkarni, and P. Scharf (Eds.): Sanskrit CL 2007/2008, LNAI 5402, pp. 1–32, 2009.

1 Padasaṁskāra and vākyasaṁskāra*

1.1 Introduction

Pāṇinīyas have considered Pāṇini's system of rules from two points of view, according to whether it serves as an explanation (*anvākhyāna*) with (A) syntactic words (*pada*)[1] or (B) utterances as its limit (*avadhi*: *padāvadhika*, *vākyāvadhika*).

The contrast between these two positions is brought out clearly in the *Vṛtti* on VP 1.24-26,[2] which concerns examples such as the following:[3]

(1) *śuklaṁ vastram* 'a white garment'
(2) *śuklā gauḥ* 'a white cow'
(3) *śuklaḥ kambalaḥ* 'a white blanket'
(4) *śuklau kambalau* 'two white blankets'

---

* I am grateful to participants in the second Sanskrit Computational Linguistics Symposium for their discussion after my presentation of this paper on May 15, 2008 as well as to Peter Scharf for discussions at other times. I welcome this opportunity to present an overview of Pāṇini's derivational system with particular emphasis on its unity and lack of modularity. This is especially pertinent currently in view of work being done in the framework of what is called distributed morphology, which brings modern theoreticians closer to what Pāṇini envisioned; see David Embick and Rolf Noyer, 'Distributed morphology and the syntax/morphology interface', in *The Oxford Handbook of Linguistic Interfaces* (ed. G. Ramchand and C. Reiss, Oxford/New York, 2007), pp. 289-323. For Pāṇini's derivational system, see PWT 136-400; 'Old Indic Grammar', in *Morphologie/Morphology: Ein internationales Handbuch zur Flexion und Wortbildung/An International Handbook on Inflection and Word-formation* (ed. Geert Booij, Christian Lehmann, Joachim Mugdan), Berlin-New York: Walter de Gruyter, 1. Halbband/Volume 1, article 5, pp. 41-51 (in particular, pp. 45-49); 'Pāṇini', in *History of the Language Sciences/Geschichte der Sprachwissenschaften/ Histoire des sciences du langage* (ed. Sylvain Auroux, E. F. K. Koerner, Hans-Josef Niederehe), Berlin-New York: Walter de Gruyter, Volume 1/1. Teilband/Tome 1, V, article 17, pp. 113-124 (especially pp. 115-20). Some of the points I bring out here were presented earlier in a lecture at Brown University on October 11, 2007 and at a conference in Delhi, November, 2004, where I concentrated on materials dealt with in §§2.2.2.2-2.2.2.3 of the present paper and on how ancillary rules apply. An abstract of the latter ('Some questions on Pāṇini's derivational system') has appeared in *Proceedings of International Symposium on Machine Translation, NLP and TSS* (*iSTRANS-2004*), November 17- 19, 2004 (ed. R. M. K. Sinha and V. N. Shukla), New Delhi: Tata McGraw-Hill, volume I, p 3.

[1] In this context, a pada is a term with a nominal or verbal ending, in accordance with A 1.4.14: *suptiṅantam padam* (PWT 23 [**49**]).

[2] VPVṛ. 68.5-6: *keṣāṁcit padāvadhikam anvākhyānam vākyāvadhikam ekeṣām* | Bhartṛhari goes on to consider consequences of the first position; see below with note 8. I have used *padasaṁskāra* and *vākyasaṁskāra*, which have currency among Pāṇinīyas; for example, see notes 30, 31. Bhartṛhari himself uses *padasaṁskāra* (*padasaṁskārahetuḥ*: VPVṛ. 2.42 [211.1-2]).

[3] (1), (3)-(5) are cited in the *Mahābhāṣya* on A 5.2.94: *tad asyāsty asminniti matup* (see PWT 241-42 [**346**]), in the course of a discussion concerning the difference between the use of terms called *guṇavacana* ('quality expression') with reference to things in which qualities reside and to qualities themselves, as in example (5); see notes 4, 6. I have substituted *gauḥ* in (2) for Patañjali's *śāṭī* 'piece of cloth, woman's garment' in order to simplify the discussion of derivates.

(5) *śuklāḥ kambalāḥ* 'several white blankets'
(6) *paṭasya śuklaḥ* 'the white of the cloth'[4]

As can be seen from these examples and as Patañjali points out (see note 4), *śukla-* is used to designate both something which has as a property the color white and the color itself, distinct from the thing in which it inheres. In (1)-(5), different nominative forms of *śukla-* 'white'[5] are used with reference to the things signified by the nominatives *vastram, gauḥ, kambalaḥ, kambalau*, and *kambalāḥ.* In (6), on the other hand, *śukla-* (nom. sg. m. *śuklaḥ*) refers to the quality (*guṇa*) that is related to a piece of cloth, referred to by the genitive singular *paṭasya*, in which this quality resides (*āśrita*)[6] through the relation of inherence (*samavāya*).[7] Considered by itself, a term

---

[4] Kātyāyana and Patañjali consider the formal differences in (1)-(5)—with *śāṭī* in (2)—during discussions of several sūtras. For example, Bh. II.394.11, 13-15/IV.161-62 (on A 5.2.94): *dṛśyate vyatirekaḥ | tadyathā paṭasya śukla iti | ... evaṁ ca kṛtvā liṅgavacanāni siddhāni bhavanti: śuklaṁ vastram śuklā śāṭī śuklaḥ kambalaḥ śuklau kambalau śuklāḥ kambalā iti | yad asau dravyaṁ śrito bhavati guṇas tasya yal liṅgaṁ vacanaṁ ca tad guṇasyāpi bhavati |* The discussion concerns whether or not a quality such as white can be said to exist separately from a thing. Kātyāyana and Patañjali conclude that they are distinct entities, and they note that a quality has the number and gender of the thing in which it rests. Elsewhere (e.g., Bh. I.228.20-21/II.104 [on A 1.2.52]), this is formulated as follows: *guṇavacanānāṁ śabdānām āśrayato liṅgavacanāni bhavanti* 'Quality words have genders and numbers according to the substrate (of the qualities).' See §1.2.3 and Peter M. Scharf, *The Denotation of Generic Terms in Ancient Indian Philosophy: Grammar, Nyāya and Mīmāṃsā* (Transactions of the American Philolosophical Society 86.3, Philadelphia: American Philosophical Society, 1996), pp. 73-74. The principle is considered more widely applicable. Thus, to account for duals and plurals such as *kumāryau, kumāryaḥ* in addition to a singular *kumārī* 'young girl' under the view that according to A 4.1.3: *striyām* (PWT 66 [**107**]) lets feminine affixes be introduced on condition that femininity is to be signified, so that this property is the meaning of an affix, Kātyāyana (4.1.3.vt. 6: *guṇavacanasya cāśrayato liṅgavacanabhāvāt*) invokes this: the property denoted by the feminine affix ***ṅīp*** takes on the gender and number of what the base *kumāra-* signifies.

[5] Respectively neuter singular, feminine singular, and masculine singular, dual and plural; on the affix in *śuklā*, see note 32.

[6] One way of accounting for the use of a quality word like *śukla-* to signify such a thing is suggested by Kātyāyana (5.2.94 vt. 3: *guṇavacanebhyo matupo luk*): in usages such as (1)-(5), *śukla-* is treated as a shortening for *śuklamat-* (→ *śuklavat-*: A 8.2.9: *mād upadhāyāś ca mator vo'yavādibhyaḥ*, PWT 347-48 [**540**]), with the taddhita suffix *mat**up*** according to A 5.2.94 (see note 3). This suffix is considered to occur potentially and disallowed—technically, replaced by zero—so that *śukla-*, now equivalent to *śuklavat-*, signifies something in which the color white occurs. As Patañjali acknowledges in his discussion of A 2.1.30: *tṛtīyā tatkṛtārthena guṇavacanena* (PWT 213 [**308**]), this is a mechanism accounting for the fact that terms such as *śukla-* and *kṛṣṇa-* are used to denote things which have the colors white and black in them, that *śuklaḥ* and *kṛṣṇaḥ* are equivalent to *śuklaguṇaḥ* '... which has the quality white in it', *kṛṣṇaguṇaḥ* '... which has the quality black in it'. Bh. I.385.7-10/II.590-91: *iha tṛtīyā tatkṛtārthena guṇenetīyatā siddham | so'yam evaṁ siddhe sati yad vacanagrahaṇaṁ karoti tasyaitat prayojanam: evaṁ yathā vijñāyeta guṇam uktavatā guṇavacaneneti | katham punar ayaṁ guṇavacanaḥ san dravyavacanaḥ sampadyate | ārabhyate tatra matublopo guṇavacanebhyo matupo lug iti | tad yathā śuklaguṇaḥ śuklaḥ kṛṣṇaguṇaḥ kṛṣṇaḥ ....* Patañjali argues that instead of using *guṇavacanena* in A 2.1.30, Pāṇini could accomplish the same results with the shorter

for like *śukla-* signifies a quality (*guṇa*), specifically the color white, which can be found in any white thing, as well as white things themselves. To refer to any such thing, moreover, one can appropriately use

(7) *śuklam* 'something white'

a neuter singular. This usage is brought into play for considering positions (A) and (B).

1.2 Padasaṃskāra

1.2.1

Under (A), having abstracted a base *sukla-* from expressions such as (1)-(5) (§1.1), a grammarian makes it his task simply to describe the formation of individual padas such as *śuklam, śuklā, śuklaḥ*. As Bhartṛhari brings out, however, this is not a simple task once position (A) is taken strictly.[8] Under this view, a term like *śukla-* is first considered as a separate independent word signifying any white thing in general, unrelated to a particular thing signified by another word with which it might be used. Since in all instances the same phonic elements such as *śukla* occur (*śrutyabhedāt* 'because of the identity of the audible terms'), such terms enter into derivation to

---

dhe sati yad vacanagrahaṇaṁ karoti tasyaitat prayojanam: evaṁ yathā vijñāyeta guṇam uktavatā guṇavacaneneti | katham punar ayaṁ guṇavacanaḥ san dravyavacanaḥ sampadyate | ārabhyate tatra matublopo guṇavacanebhyo matupo lug iti | tad yathā śuklaguṇaḥ śuklaḥ kṛṣṇaguṇaḥ kṛṣṇaḥ .... Patañjali argues that instead of using *guṇavacanena* in A 2.1.30, Pāṇini could accomplish the same results with the shorter formulation using *guṇena* alone, and that this shows that by *guṇavacanena* he refers to a term which signifies first a quality, then a thing in which a quality resides. It is not possible to enter into more details here. Suffice it to say that I agree to the extent that *guṇavacana* is indeed used with reference to terms that signify a thing in which a given quality occurs. The alternative to Kātyāyana's way of accounting for this is simply to accept that by Pāṇini's time grammarians had come to consider either that bases like *śukla-* had two meaings or that one had to operate with homophonous bases, one with a fixed gender and number designating a quality, the other with variable gender and number referring to things.

7 In turn, the color is itself the locus of a property, that of being white (*śuklatva*), which resides in the color by inherence. This abstract property also inheres in the cloth that is the locus of the color, but indirectly, through inhering in the color that inheres in the cloth (*samavetasamavāya*).

8 VPVṛ. 68.6-69.2 *tatra padāvadhike'nvākhyāne śrutyabhedād ekapadarūpopagrahe sāmānyamātre labdhasaṁskārāṇi padāni padāntarasambandhaprāptasannidhāneṣv apy artheṣu sannipatiteṣv api viśeṣeṣu sāmānye pratilabdham antaraṅgaṁ saṁskāram upādāyaiva pravarteran | tatraikavacanānto napuṁsakaliṅgaś ca śuklaśabdo bhinnaliṅgasaṅkhyair āśrayaih sambaddhaḥ śrūyeta | tadarthaṁ viśeṣanānāṁ cājāteḥ* (A 1.1.52) *ity anena yogena bhāviny āśraye bahiraṅge prakrānte guṇavacanānāṁ śabdānām āśrayato liṅgavacanāny anugamyante | vākyāvadhike tv anvākhyāne nityasaṁsṛṣṭasya guṇasyāśrayaviśeṣeṇātyantam avivekāt sarvato vyavacchede sāmānyārthatvam eva na vidyate |*

form words[9] used with reference solely to what is general (*sāmānyamātre*),[10] that is, any substance in which the qualities in question reside, so that they can take designations *śukla* and so on.[11] Accordingly, the meaning signified by *śukla* or a comparable term like *kṛṣṇa* 'black' is not now characterized by any particular gender or number.[12] Yet in examples such as (1)-(5), the meaning of *śukla-* is indeed associated with different genders and numbers, and this has to be accounted for.

If one stays strictly within the confines of position (A), terms like *śukla-* will first form padas by taking only an ending that is appropriate to the indeterminate meaning, as in (7). Further, since the position is taken that padas are derived as separate units, a pada thus formed has primary status. Schematically, then, *śuklam* in (7) can be considered to be surrounded by brackets showing that it is an independent unit: (*śuklam*).

---

[9] The *Vṛtti* (68.6-7, see note 8) uses the phrase *labdhasaṁskārāṇi padāni* 'words that have received their saṁskāra'. That is, the terms in question are padas according to A 1.4.14 (see note 1), so that they have received a grammatical explanation in terms of a derivation which involves introducing a nominal ending after a base. In (1)-(6), *śukla-* is followed by endings of the first triplet (*prathamā*) of nominal endings, introduced on condition that the meaning of the base as well as gender and number are to be signified, according to A 2.3.46 (PWT 156 [**240**]): *prātipadikārthaliṅgaparimāṇavacanamātre prathamā.*

[10] *ekapadarūpopagrahe sāmānyamātre* of the phrase *śrutyabhedād ekapadarūpopagrahe sāmānyamātre labdhasaṁskārāṇi padāni* is subject to different interpretations, depending on how one understands the compound *ekapadarūpopagraha* of *ekapadarūpopagrahe*. This can be understood as a ṣaṣṭhītatpuruṣa (*ekaṁ padarūpam ekapadarūpam, ekapadarūpasyopagrahaḥ*) containing an action noun *upagraha* and meaning 'the grasping of a single word form'. Under this interpretation, *ekapadarūpopagrahe* of *śrutyabhedād* is most simply understood as a locative absolute (*sati saptamī*): 'there being a grasping of a single word form due to the non-difference in phonic form'. This interpretation is explicitly adopted by Raghunātha Śarmā (VPA 1.14-26 [62.24-26]: *śuklatvarūpasya sāmānyasya guṇe guṇini ca yathākramaṁ samavāyena svāśrayasamavāyena ca sattvād iti tadānīṁ dvayor api śuklaśabdayor varṇānupūrvīśrutyabhedād ekapadarūpatvopagrahe sati śuklatvarūpaṁ sāmānyam āśritya labdhasaṁskāraṁ dvividham api śuklapadam* |), who considers the common property (*sāmānya*) in question to be the generic property of being white (*śuklatva*, see note 7) and the identity of phonic form to hold between the two terms *śukla*, one denoting this generic property, the other referring to a white thing. In an alternative interpretation, *ekapadarūpopagrahe* is a bahuvrīhi qualifying *sāmānyamātre*. The constituent *upagraha* is now an instrument noun denoting a means of grasping and this noun is coreferential with *ekapadarūpa* (*ekapadarūpam upagraho yasya tat sāmānyamātram*). *ekapadarūpopagrahe sāmānyamātre* now refers to something purely general (*sāmānyamātra*) for the grasping of which the single word form serves as a means. Śrīvṛṣabha appears to have understood this, since he notes that, due to the phonic identity, it is possible to grasp a general referent by means of a single word form: *yad āha: śrutyabhedād iti* | *ataś caikenaiva padarūpeṇopagrahītuṁ śakyate* | *sāmānyamātre iti* ... (VPP 1.24-26 [69.10-11]). In addition, he takes *sāmānya* to refer to white things in general; see note 11. Whether *sāmānya* is considered here to refer to a general group of things sharing a common property or to a generic property does not affect the ultimate import. In this context, it is worth bringing in A 2.1.55: *upamānāni sāmānyavacanaiḥ*. According to this

This is now not related to any other word like *vastram* so that the grammatical process of introducing an ending after *śukla-* and other operations that apply to form *śuklam*[13] can be qualified as internally conditioned (*antaraṅga*). Further, even when the color spoken of is related to other entities, so that it thereby comes to be juxtaposed with them (*padāntarasambandhaprāptasannidhāneṣv api*) and the particular white things are spoken of together with this color (*sannipātiteṣv api viśeṣeṣu*), as in (1)-(5), the padas of these sentences are considered to have received their grammatical formation (*labdhasaṁskārāṇi*) solely with respect to the common quality (*sāmānyamātre*)—a white color. Moreover, this formation is now considered to be internally conditioned.

---

sūtra, padas which signify things to which something else is compared (*upamāna*) form tatpuruṣa compounds with related padas signifying what is common (*sāmānyavacana*); a standard example is *śastrī-śyāmā-*, as in *śastrīsyāmā devadattā* 'Devadattā is black as a knife', equivalent to *śastrīva śyāmā devadattā*. The constituent *śyāma-* of the compound refers to some black thing, not to the generic property of being black (*śyāmatva*), although it is this property that is common to both the knife and to the woman being compared to it. Accordingly, the *Kāśikā* remarks *upamānopameyayoḥ sādhāraṇo dharmaḥ sāmānyam* | *tadviśiṣṭopameyavacanair ayaṁ samāsaḥ* 'What is common is the property common to the standard of comparison and what is compared to it; this compound (is formed) with padas signifying things that are qualified by this property and are being compared' (Kāś. 2.1.55). Jinendrabuddhi appropriately invokes the parallel between the terms *sāmānyavacana-* and *guṇavacana-* (see note 6), remarking (N 2.1.55 [II.72.26-27: *sāmānyam uktavantaḥ sāmānyavacanāḥ yathā guṇam uktavanto guṇavacanā iti* | *kadā ca te sāmānyavacanā bhavanti* | *yadā sāmānyam abhidhāya sāmānyaviśiṣṭe tadvati dravye vartante* | similarly PM II.72.8-10) that, as *guṇavacana-* is used of terms which first signify a property and then a thing with that property, so *sāmānyavacana* is used of terms which first signify a general common property and then are used with reference to some thing which is qualified by such a property.

[11] Śrīvṛṣabha uses *āśrayasāmānya* 'common substrate', explaining as follows. When such a word signifies a substance primarily and a quality only subordinately with respect to a substance that is the principal significand, then it has the status of being a qualifier; if, on the other hand, such a term signifies a quality by itself, it does not have this status. Further, when guṇavacana terms like *śukla* are used to signify things, even though there may be a particular thing, the terms themselves—considered as equivalent to derivates with *mat**up***—refer merely to some indeterminate things which have qualities such as the color white. Moreover, since under the position adopted, there is no relation with the meaning signified by any other word—such as *vastram* in (1)—*śukla* and so on have an undifferentiated status (*abhinnarūpatā*) because they designate any common substrate (*āśrayasāmānyābhidhānāt*) of the color. VPP 1.24-26 (69.5-9): *śuklādayaḥ śabdā guṇavṛttitve'pi yadā guṇopasarjanadravyaṁ matublopāt pratipādayanti tadāyam upanyāsaḥ* | *tadā hy eṣāṃ guṇopasarjanadravyābhidhānād viśeṣaṇatvam iti* | *guṇasya svātantryeṇābhidhāne na viśeṣaṇatvam* | *tadā ca yady api viśeṣo vartate tathāpy evaṁ padagatam artham āśritya śuklo guṇo'syāstīti vṛttiśabdapadāntarapratipādyaviśeṣāsaṁsparśād abhinnarūpatā teṣu vartate āśrayasāmānyābhidhānāt* |

[12] To be sure, *śukla-* in (6) *paṭasya śuklaḥ* has a fixed gender, masculine, but this refers to the color white distinct from the cloth which has this color; see note 4.

[13] The ending ***su*** of the first triplet (see note 9) is introduced after *śukla-*, because the meaning in question is characterized by singularity (A 1.4.22: *dvyekayor dvivacanaikavacane*, PWT

Therefore, the padas in question would enter into play without giving up this internal formation.[14] Consequently, one would now allow a neuter singular form *śuklam* in (1)-(5) instead of different forms in accordance with the particular referents.

1.2.2

Obviously, this consequence cannot be accepted. Bhartṛhari goes on[15] to say that it is for this reason—that is, to avoid the undesired consequences—that the sūtra A 1.2.52: *viśeṣaṇānāṁ cājāteḥ*[16] is formulated. Under the view adopted, this sūtra has the following effect: gender and number (*liṅgavacanāni*) for terms that signify qualities and things in which these qualities reside (*guṇavacanānām*) are understood[17] to be in accordance with the particular locus (*āśrayataḥ*) of the quality in question; that is, terms signifying qualities of things do so with the gender and number ascribed to the substances in which the qualities reside. Bhartṛhari also notes that this applies once a future locus has come to the fore, which is external: *bhāviny āśraye bahiraṅge prakrānte*. That is, although *śukla* is first considered independently, without regard to

---

152, 157 [**234**, **241**]). Since the meaning is also characterized as neuter, ***su*** is then replaced by *am* (A 7.1.24: *ato'm*, PWT 323 [**495**]): *śukla-s* → *śukla-am* → *śuklam* (A 6.1.107: *ami pūrvaḥ*, PWT 344 [**533**]). The basis for the neuter gender and for selecting the singular ending in such examples cannot be discussed here. The point of view Bhartṛhari brings up may well have been adopted by some grammarians much earlier. For, in the *Mahābhāṣya* on A 1.1 1 (Bh. I.39.17-19/I.129), Patañjali remarks that padas are formed and then connected as one wishes (see PWT 142-43 [**225**]); however, in the context, he is speaking in particular of word order, so that he need not be referring to the point of view which Bhartṛhari is considering.

14 *sāmānye pratilabdham antaraṅgaṁ saṁskāram upādāyaiva pravarteran* 'would enter into play only after receiving the internally conditioned formation they have gotten with reference to what is common.' Śrīvṛṣabha (VPP 69.17: *upādāyaiva iti* | *aparityajya*) glosses *upādāyaiva* with *aparityajya* 'without giving up'.

15 VPVṛ. 1.24-26 (68.9-69.1): *tadarthaṁ viśeṣaṇānāṁ cājāterity anena yogena bhāviny āśraye bahiraṅge prakrānte guṇavacanānāṁ śabdānām āśrayato liṅgavacanāny anugamyante* | In a different context, Bhartṛhari (VP 3.14.135: *abhede liṅgasaṅkhyābhyāṁ yogāc chuklaṁ paṭā iti* | *prasakte śāstram ārabdhaṁ siddhaye liṅgasaṅkhyayoḥ* ||) brings this issue up again. He notes that if there is no distinction by gender and number to begin with, this would allow utterances like *śuklaṁ paṭāḥ*, with a neuter singular in construction with a masculine plural, and that A 1.2.52 is formulated in order to establish the correct gender and number in such instances. He also later (VP 3.14.140: *bhāvino bahiraṅgasya vacanād āśrayasya ye* | *liṅgasaṅkhye guṇānāṁ te sūtreṇa pratipādite* ||) says that this sūtra provides for the gender and number of an external substrate of qualities related to these qualities. In the present paper, it is not necessary to discuss this further.

16 A 1.2.52 forms a pair with A 1.2.51: *lupi yuktavadvyaktivacane* (PWT 593-94 [**867**]). This rule provides for the number and gender (*vyaktivacane*) of derivates in which a taddhita affix has been replaced by the particular zero designated *lup*. The meaning of the derivate has

any possible qualificand, one provides by rule that such a quality word takes the affixes appropriate to the gender and number of qualificands denoted by terms with which it will be used.

1.2.3

Interpreted in this manner, A 1.2.52 serves the same purpose as

(8) *guṇavacanānāṁ śabdānām āśrayato liṅgavacanāni bhavanti* (see note 4)

and Patañjali ends the first part of his discussion on A 1.2.52 by concluding that this sūtra accomplishes the purpose served by (8), which thereby does not have to be formulated.[18] The short discussion which leads to this conclusion concerns the purpose for which A 1.2.52 is stated.[19] At issue is the use of terms that signify things qualified by generic properties (*jāti*) found in all members of a class of things and terms signifying qualities (*guṇa*), respectively called *jātiśabda* ('generic word') and *guṇaśabda* ('quality word'). The main point of contention concerns what purpose A 1.2.52 serves. This rule can serve two purposes, one positive, the other negative. As a complement to A 1.2.51, it can add that the gender and number of a base is extended not only to the meaning of a derivate in which a taddhita suffix has been replaced by the zero called *lup* (see note 16) but also to its qualifiers. In addition, as a restriction to this provision, the rule can exclude this extension for a generic term like *janapada* 'district'.

In his first vārttika, Kātyāyana appropriately links this rule to the preceding sūtra, A 1.2.51. According to him, A 1.2.52 is stated in order to keep a generic term serving as a qualifier from having the gender and number assignment that would obtain by A 1.2.51.[20] For example, *janapadaḥ* in

(9) *pañcālā janapadaḥ* 'the Pañcāla district'

is singular, but *ramaṇīyāḥ* of

---

the gender and number of the meaning of the base whose affix has been replaced by *lup*. For example, *pañcālāḥ* refers to a district (*janapada*) inhabited (at least originally) by the Pañcālas, and the derivate *pañcāla-* denoting this district has the gender and number—masculine and plural—which hold for the meaning of the base *pañcāla-* that denotes the Pañcāla people. *lupi* is usually interpreted as a locative absolute (*lupi sati*) and *yuktavat* as a form with the suffix *vati* (A 5.1.116: *tatra tasyeva*, PWT 242 [**347**], see §2.2.2.2) equivalent to *prakṛtivat* 'as of the base' (thus, e.g., RA I.239, PK I.763, SK II.377). I accept these as the most justified interpretations, although the *Kāśikā* invokes others, which need not be considered here.

17 *anugamyante*; Raghunātha Śarmā (VPA 1.24-26 [63.12]) suggests that this means 'are taught' (*anuśiṣyante*); this detail is of no consequence to the general discussion.

18 Bh. 1.2.52 (I.228.19-21/II.104): *na tarhīdānīm ayaṁ yogo vaktavyaḥ | vaktavyaś ca | kiṁ prayojanam | idaṁ tatra tatrocyate: guṇavacanānāṁ śabdānānām āśrayato liṅgavacanāni bhavantīti | tad anena kriyate |*

(10) *pañcālā ramaṇīyāḥ* 'The Pañcāla is pleasant'[21]

is plural. Similarly, *vṛkṣaḥ* 'tree' in

(11) *badarī vṛkṣaḥ* 'badarī tree'

is masculine. Were A 1.2.51 allowed to hold in all cases, one would allow a plural *janapadāḥ* also in (9) and a feminine **vṛkṣā* in (11).

Kātyāyana's first vārttika presupposes that the number and gender associated with the qualifier denoted by *ramaṇīya-* of *ramaṇīyāḥ* in (10) is not provided for by A 1.2.52, which therefore applies only with respect to generic terms. Accordingly, the Bhāṣya then questions why Kātyāyana's first vārttika says what it does and why Kātyāyana does not also say that the sūtra is stated so that qualifiers get the gender and number of bases.[22] The reasoning behind Kātyāyana's first statement is then brought out in his second vārttika:[23] there is no need for this positive provision because, by virtue of qualifier terms being coreferential (*samānādhikaraṇatvāt*) with terms denoting qualificands, the required result is established (*siddham*), so that *ramaṇiya-* in (10) and other terms signifying qualifiers will be treated as having the gender and number of *pañcāla-* and so on by A 1.2.51, without the need for another rule. Patañjali immediately objects that if this is so, then A 1.2.52 serves no purpose. For, he points out,[24] in usages other than where an affix is replaced by *lup* (see note 16), a generic does not take on the gender and number of the particular thing it qualifies.[25] For example, in

(12) *badarī sūkṣmakaṇṭakā madhurā vṛkṣaḥ* 'The badarī is a soft-thorned, sweet tree'

sūkṣmakaṇṭakā ('with soft thorns') and *madhurā* ('sweet') are feminine, in agreement with *badarī*, but the generic term *vṛkṣaḥ* is masculine.[26] This being the case, it is useless to single out one type of usage involving generic terms and having A 1.2.52 apply with respect to them alone.

This does not mean, however, that in Patañjali's estimation A 1.2.52 serves no purpose whatever. As shown, he considers that this sūtra serves the same purpose as the agreement rule (8).

---

[19] Bh. 1.2.52 (I.228.11/II.103): *kimarthaṁ punar idam ucyate* |

[20] 1.2.52 vt. 1: *viśeṣaṇāṁ vacanaṁ jātinivṛttyartham* |

[21] 'The Pañcāla' is to be understood in the same way as a phrase like 'the Panjab'.

[22] Bh. 1.2.52 (I.228.13-14/II.103): *jātinivṛttyartho'yam ārambhaḥ* | *kim ucyate jātinivṛttyartha iti na punar viśeṣaṇānām api yuktavadbhāvo yathā syād iti* |

[23] 1.2.52 vt. 2: *samānādhikaraṇatvāt siddham* | Bh. I.228.16/II.103: *samānādhikaraṇatvād viśeṣaṇānāṁ yuktavadbhāvo bhaviṣyati* |

[24] Bh. 1.2.52 (I.228.16-18/II.103): *yady evaṁ nārtho'nena* | *lupo'nyatrāpi jāter yuktavadbhāvo na bhavati* | *kvānyatra* | *badarī sūkṣmakaṇṭakā madhurā vṛkṣa iti* |

[25] I have paraphrased *jāter yuktavadbhāvo na* reflecting Nāgeśa's (Ud. II.103/II.64a: ... *jāter yuktavadbhāvo neti* | *viśeṣyaliṅgādigrāhitā nety arthaḥ*) apt paraphrase.

[26] Patañjali (Bh. I.228.18-19/II.103-4: *kiṁ punaḥ kāraṇam anyatrāpi jāter yuktavadbhāvo na bhavati* | *āviṣṭaliṅgā jātir yal liṅgam upādāya pravartate utpattiprabhṛty ā*

1.2.4

The question remains, nevertheless, whether A 1.2.52 indeed can serve no purpose other than denying for generic entities the properties provided for in the preceding sūtra. Let us consider more carefully on what grounds one can maintain the assumption made in Kātyāyana's second vārttika on A 1.2.52 (§1.2.3 with note 23). The derived base *pañcāla-* of *pañcālāḥ* in (9), (10) (§1.2.2) is formed with the taddhita affix ***aṇ*** introduced after *pañcāla-ām: pañcāla-ām-a- → pañcāla-a- → pañcāl-a-*.[27] In addition, the taddhita affix in question here has the meaning 'abode, dwelling place (*nivāsa*)', and *pañcālāḥ* is equivalent to the string

(13) *pañcālānāṁ nivāsaḥ* 'dwelling place of the Pañcālas'.

In terms of its semantics, then, *pañcāla-* in (9) and (10) should be singular, since the derivate in question denotes an abode, a district where the Pañcālas dwell. In order to account for the plurality of the derivate, then, A 1.2.51 transfers to the meaning of the taddhita suffix which is deleted the number property of the base with which this affix first occurs, that is, the plurality proper to the Pañcālas who inhabit a group of villages. Now, a qualifying term such as *ramaṇīya* can be used coreferentially with *pañcāla* that results from the derivation noted. These are coreferential in that *ramaṇīya* refers to something pleasing, a district, and *pañcāla* also refers to a district, which is the meaning of the suffix ***aṇ*** that has been deleted. This does not mean, however, that *ramaṇīya* now also refers to this affix meaning characterized by the plurality transferred to it by A 1.2.51. If this were the case, then, A 1.2.52 would indeed not serve the purpose of providing that a qualifier also has the number proper to a base meaning.[28] On the other hand, if A 1.2.51 serves to transfer gender and number of bases only to the meaning of a suffix in a derivate which itself results from deleting this suffix, then, it does not of itself automatically allow a coreferential term to have these properties merely by virtue of its also referring to that affix meaning. Under this condition, A 1.2.52 does serve a purpose in transferring to such qualifiers the gender and number of a base meaning. An exception is then made for generic terms.[29]

---

vināśān na tal liṅgaṁ jahāti |) goes on to explain why a generic behaves this way: from the time a generic property first enters into play as something denoted by particular terms, it keeps and does not give up this gender up to whatever time one ceases to speak of it. That is, although generic terms such as *janapada* and *vṛkṣa* can denote qualifiers of other entities, they are always associated with a fixed gender. See Scharf, op. cit. (note 4), pp. 74-75.

[27] See PWT 591 ([**862**]).

[28] Jinendrabuddhi makes this point explicit: *yady api bhinnaṁ viśeṣaṇānāṁ pravṛttinimittaṁ tathāpi tena bhinnenāpi tatra pravartamānānāṁ sa eva lubartho 'tidiṣṭavyaktivacanas teṣām abhidheyaḥ* | *tasmāt tadgatenaiva liṅgasaṅkhyābhidhāne siddhe jātipratiṣedhārtham idaṁ vacanam* | (N 1.2.52 [I.356.27-30]).

[29] This conclusion assumes that Pāṇini formulated A 1.2.52 with a purpose and that this rule is immediately connected with the preceding sūtra. I have also stressed the fact that A 1.2.51

Consequently, A 1.2.52 need not serve merely as an equivalent to (8) (§1.2.3). It can have a purpose of its own, associated with the extension of gender and number provided for specifically in the rule with which it is paired. Thus, A 1.2.52 is not simply a rule providing for gender and number agreement between terms signifying qualifying properties and qualificands. In order to avoid the undesired consequence of allowing a neuter singular such as *śuklam* to occur in construction with *śāṭī* and so on in examples like (2)-(5) (see §§1.1, 1.2.1), then, under position (A), one must now explicitly state the agreement rule (8).[30]

1.3 Vākyasaṁskāra

After outlining position (A) and its difficulty, Bhartṛhari goes on (see note 9) to contrast what holds under (B) (*vākyāvadhike tv anvākhyāne*). Now a quality such as white is always fused (*nityasaṁsṛṣṭasya guṇasya*) with a particular substrate (*āśrayaviśeṣeṇa*) such as a piece of cloth. It is thus never the case that a quality occurs apart from such a substrate (*atyantam avivekāt*). Hence, each instance of a quality, necessarily fused with a substrate, is set apart from every other instance of such a quality (*sarvato vyavacchede*), which also always is fused with a particular thing. This

extends gender and number properties to the meaning of the deleted affix, so that this extension applies narrowly: when the meaning in question is denoted by the presuffixal base. The view that part of A 1.2.52 serves no immediate purpose assumes that A 1.2.51 extends the gender and number of a base to the meaning of the affix which is replaced by *lup* and does so in general, so that the extension applies with respect to that meaning even if it is not denoted directly by the presuffixal base.

[30] Kaiyaṭa points this out: under the padasaṁskāra view, (8) is to be stated explicitly (*vācanikam*). For, if a pada is formed without regard for any other pada, then the general neuter (*napuṁsakaṁ liṅgasarvanāma* 'the neuter that is the pronoun among genders') and a singular that does not depend on any other thing obtain, so that an utterance like *śuklaṁ paṭāḥ* would be allowed. This being possible, (8) conveys that the gender and number apply which are connected with some external substrate that will occur. Pr. II.104/II.64b: *guṇavacanānām iti* | *padasaṁskārapakṣe vācanikam etat* | *pade hi padāntaranirapekṣe saṁskriyamāṇe napuṁsakaṁ liṅgasarvanāma prāptam ekatvaṁ ca vastvantaranirapekṣatvāt sannihitam iti śuklaṁ paṭā iti prāpte bhāvino bahiraṅgasyāśrayasya sambandhinyau liṅgasaṅkhye anena pratipādyete* | In connection with the possible singular, Śrīvṛṣabha (VPP 1.24-26 [69.12-13) invokes the singularity associated with non-differentiation (*abhedaikatvasaṅkhyā*), which is justified because no specific referent is in play: *āśrayasāmānyarūpasya ca liṅgasarvanāmnā napuṁsakena yogo vyaktiviśeṣaviṣayatvāt saṁstyānādīnām abhinnatvāc cābhedaikatvasaṅkhyeti talliṅgasaṅkhye kutaḥ* | This undifferentiated singular is usually associated with constituents of compounds (see §2.2.3.1), but is not necessarily limited to these, as is clear from what Śrīvṛṣabha says. There is a recent study, limited to compounds, of this concept by Pascale Haag: 'Du nombre grammatical dans les composés Sanskrits: le concept d'*abhedakatvasaṃkhyā*', Histoire Épistémologie Langage 27/1: 127-52 (2005).

being so, a quality simply does not have the property of being a general meaning (*sāmānyārthatvam eva na vidyate*). Under position (B), padas of utterances like (1)-(5) (§1.1) are thus considered to be formed with affixes following bases whose meanings are related from the outset.

If this position is adopted, (8) does not serve to provide for gender and number agreement. This statement now merely reiterates what holds by the very nature of the relationships.[31] Thus, the nominal base *śukla-* of forms in (1)-(5) denotes white color as it is located in the particular substances denoted by the bases *vastra-*, *go-*, *kambala-*. These nominals and *śukla-* are now coreferential (*samānādhikaraṇa*) in that, for example, *śukla-* of (1) refers not to just anything that is a locus of the color white but a garment that is white, and *vastra-* here refers not to just any member of the set of possible garments—all characterized by the property of being a garment (*vastratva*)—but to a white one in particular. Similarly, *go-* and *śukla-* are corefential. Accordingly, the latter refers to a thing which has the property of being feminine (*strī*), so that it takes the feminine suffix ***ṭāp***.[32] In all the examples, the base meanings along with genders are signified, so that endings of the first triplet are introduced (see note 9), and in accordance with the numbers of the things in question, the endings called *ekavacana*, *dvivacana*, and *bahuvacana*—singular, dual, and plural endings—are allowed to occur:[33] *śukla-s* → *śukla-am* → *śuklam* (see note 13), *vastra-s* → *vastra-am* → *vastram*,[34] *śuklā-s* → *śuklā*,[35] *go-s* → *gau-s*,[36] *śukla-au* → *śuklau, kambala-au* → *kambalau*,[37] *śukla-as* → *śuklās, kambala-as* → *kambalās*.[38]

1.4 Summary

Although there were doubtless grammarians who, as Bhartṛhari notes (see §1.1), maintained that padas could be derived and then allowed to occur in larger utterances, there can equally be no doubt whatever that Pāṇini's system does not operate in this manner. On the contrary, Pāṇini's grammar sets forth a derivational system in which padas are formed through introducing affixes after verbal and nominal bases—as well as feminine derivates with suffixes designated by the abbreviatory terms ***ṅī*** and ***āp***-

---

[31] Kaiyaṭa points this out also: *yadā tu vākyasaṁskāras tadāyam anuvāda eva* | (Pr. II.104/II.64b).

[32] A 4.1.4: *ajādyataṣ ṭāp* (PWT 180 [**271**]): *śukla-ā* → *śuklā* (A 6.1.101: *akaḥ savarṇe dīrghaḥ*, PWT 343 [**532**]).

[33] A 1.4.21-22: *bahuṣu bahuvacanam* | *dvyekayor dvivacanaikavacane* (PWT 152, 157 [**234**, **241**], see note 13).

[34] *śuklam vastram* → *śuklaṁ vastram* (A 8.3.23: *mȯ nusvāraḥ*, PWT 358 [**559**]).

[35] A 6.1.68: *halṅyābbhyo dīrghāt sutisy apṛktaṁ hal* (PWT 278, 309 [**404**, **467**]).

[36] A 7.1.90: *goto ṇit*, 7.2.115: *aco ñṇiti* (PWT 332, 284 [**511**, **417**]); *gaus* → *gau-R* → *gauḥ* (A 8.2.66: *sasajuṣo ruḥ*, 8.3.15: *kharavasānayor visarjanīyaḥ*, PWT 353, 357 [**551**, **557**]).

[37] A 6.1.104: *nād ici*, 6.1.88: *vṛddhir eci* (PWT 344, 342 [**532**, **529**]).

[38] A 6.1.102: *prathamayoḥ pūrvasavarṇaḥ* (PWT 343 [**532**]); *-ās* → *-āR* → *-āḥ* (see note 36).

whose meanings are related to each other in specified ways. This is obvious from the fact that, for example, a nominal ending of the second triplet, such as *am*, is introduced after a nominal on condition that an immediate object of an action is to be signified (*karmaṇi*),[39] provided also that this is not already signified otherwise.[40] Thus,

(14) *devadattaḥ śuklāṁ gām ā nayati* 'Devadatta is leading hither (*ā nayati*) a white (*śuklām*) cow (*gām*)'

(15) *devadattena śuklā gaur ā nīyate* 'A white cow is being led hither by Devadatta'

are accounted for as alternative expressions. (15) has the instrumental singular *devadattena*, derived from *devadatta-ā*, with the third-triplet ending *ṭā*, introduced to signify an agent,[41] and the third singular passive *nīyate*, which derives from *nī-ta*, with the ending *ta* (→ *te*) signifying an object (*karman*) of leading.[42] Here, an agent has not otherwise been signified, so that an agent-signifying ending follows *devadatta-*, which refers to the particular agent involved in the action. (14), on the other hand, has the ending *ti**p*** (← ***l**a**ṭ***, see note 42), which signifies an agent. Accordingly, *devadatta-* is now not followed by an agentive ending; instead, the ending *s**u*** of the first triplet is introduced (see note 9). Moreover, since the immediate object of the action is now not signified by the verb ending in (14), the second-triplet ending *-am* is introduced after *go-*: *go-am*.[43] In both (14) and (15), the cow is qualified as white; *śuklā-* here is coreferential (*samānādhikaraṇa*) with *go-*, so that it too refers to an immediate object of the act of leading, hence also is followed by the appropriate endings: *am* in (14)[44] and *s**u*** in (15).

The application of affixing rules to introduce endings after the qualifier *śuklā-* in (14)-(15) thus depends on semantics: the qualifier word is coreferential with the term that refers to the qualificand. Pāṇini makes coreference a condition also for determining which triplet of verbal endings in the set of eighteen basic endings is to be selected to replace an L-affix (see note 42): if the L-affix is coreferential (*samānādhikaraṇe*) with a potentially (*sthāniny api*) cooccurring pronominal *yuṣmad* (2nd p. pr.) or *asmad* (1sg p. pr.), respectively, endings of the triplets called *madhyama* (*si**p** thas tha*, *thās āthām dhvam*) and *uttama* (*mi**p** vas mas*, *i**ṭ*** *vahi mahi**ṅ***) are selected, otherwise endings of the triplet called *prathama* (*ti**p** tas jhi*, *ta ātām jha*) are allowed to occur.[45]

---

[39] A 2.3.2: *karmaṇi dvitīyā* (PWT 155-56 [**240**]).

[40] A 2.3.1: *anabhihite* (PWT 155 [**240**]).

[41] A 2.3.18: *kartṛkaraṇayos tṛtīyā* (PWT 155-56 [**240**]).

[42] The ending itself derives from an abstract L-affix ***l**a**ṭ***; such affixes are introduced on condition that an agent or an object is to be signified and, if the act in question does not have an immediate object, when an agent or the action itself is to be signified: A 3.4.69, 77-78: *laḥ* (*kartari*) *karmaṇi ca bhāve cākarmakebhyaḥ* | *lasya* | *tiptasjhi* ... (PWT 148 [**232**]).

[43] *go-am* → *gām* (A 6.1.93: *auto'mśasoḥ*, PW 342 [**530**]).

[44] *śuklā-am* → *śuklām* (A 6.1.107: *ami pūrvaḥ*, PWT 344 [**533**]).

[45] A1.4.105, 107-8: *yuṣmady upapade samānādhikaraṇe sthāniny api madhyamaḥ* | *asmady uttamaḥ* | *śeṣe prathamaḥ* | (PWT 151 [**234**]).

2. Pāṇini's Derivational System as a Continuum

2.1 Introduction

Western grammarians and linguists are commonly accustomed to thinking of linguistic systems—including phonology and grammar—as compartmentalized. One thus speaks of morphology on the one hand and syntax on the other and of inflectional morphology as opposed to derivational morphology. Pāṇini's system, however, is not so sharply compartmentalized. The basic structure of his derivational system is instead a continuum, starting from meanings that a speaker is to convey through utterances.

Meanings serve as conditions (*nimitta*) for the occurrence of affixes after bases. Consider, for example, the sūtras that serve to introduce two L-affixes after verbal bases in deriving conditionals like

(16) *kr̥ṣṇaṁ namet cet sukhaṁ yāyāt* 'If (*cet*) one does obeissance (*namet* 'should bow') to Kr̥ṣṇa, one attains (*yāyāt* 'may go to') happiness (*sukham*)'

(17) *edhāṁś ced alapsyata odanam apakṣyat* 'If he had gotten (*alapsyata*) firewood (*edhān*), he would have cooked (*apakṣyat*) some rice (*odanam*).'

Both of these utterances involve a relation between two actions: one is the cause of the other, so that they are related as cause (*hetu*) and effect (*hetumat* 'which has a cause'). A rule[46] introduces the L-affix ***liṅ*** after verbal bases whose meanings are related as cause and effect (*hetuhetumatoḥ*). Thus, *namet* and *yāyāt* of (16) derive from *nam-l* and *yā-l*, with ***liṅ*** after the bases *nam* 'bend, bow' and *yā* 'go to, reach'.[47] (17) also involves two actions related in this way, with one additional feature: this is a contrafactual conditional, such that the related actions do not take place. To account for such utterances, other rules[48] introduce ***lr̥ṅ*** after verbs: *labh-l ... → alapsyata, pac-l → apakṣyat.* According to these sūtras, this L-affix follows verbs if there is the condition related to ***liṅ*** (*liṅnimitte*)—that is, if the actions in question are related as cause and effect—provided also that the actions do not actually take place (*kriyātipattau*). Clearly, the tatpuruṣa compound *liṅnimitta* (loc. sg. *liṅnimitte*) refers to the condition which determines the occurrence of ***liṅ***.[49]

---

[46] A 3.3.156: *hetuhetumator liṅ* (PWT 168 [**257**]).

[47] Details of derivational stages are not important to the present discussion; see PWT 168.

[48] A 3.3.139-40: *liṅnimitte lr̥ṅ kriyātipattau | bhūte ca* (PWT 169 [**258**]).

[49] This accords with Pāṇini's derivational system being formulated from a speaker's point of view. In accordance with this also, Patañjali remarks that a meaning should not depend on a speech form (*śabdakr̥tena* 'produced by a speech unit'); on the contrary, a speech form should depend on a meaning (*arthakr̥tena*): *na hi śabdakr̥tena nāmārthena bhavitavyam | arthakr̥tena nāma śabdena bhavitavyam* (Bh. I.362.17-18/II.510, on A 2.1.1; similarly, Bh. I.464.15-16/II.822-23, on A 2.3.50, III.253.13-14/V.37, on A 7.1.33). That is, from the speaker's viewpoint, a speech unit is used to express an intended meaning, which thus determines what a speaker says.

2.2 Morphology and syntax[50]

2.2.1 Declensional and Conjugational Morphology

Just as clearly as verbal affixes occur under conditions that involve related actions in the Pāṇinian system, the verbal and nominal endings introduced in deriving utterances such as (14)-(15) (§1.4) and (16)-(17) (§2.1) are considered to occur under conditions that involve relations among the meanings of verbal bases and nominal items. Of course, not all affixation involves relations between actions and participants in them. There are also relations that hold between the denotata of nominals, for example the relation between a father and his son, as when one says

(18) *rāmasya pitā* 'Rāma's father'.

Nevertheless, here also there is a relation involving two entities, and a sixth-triplet ending occurs to signify such a relation.[51] In addition, affixes are introduced under cooccurrence conditions. For example, endings of the third triplet follow a nominal that is syntactically connected or connectable with *saha*,[52] as in

(19) *putreṇa saha* 'with his son'.

In sum, what are commonly called inflectional morphology and verbal conjugation are part and parcel of syntactic derivation. There is no question of dealing with padas except as they are related to others.[53]

2.2.2 Derivational Morphology

2.2.2.1 Kṛt Affixes. Affixes called *kṛt* in Pāṇini's system, which form what modern grammars call primary derivates, also are introduced in syntactic derivations. Consider

(20) *devadattaḥ kaṭam akārṣīt* 'Devadatta (has) made a mat'
(21) *devadattaḥ kaṭaṁ kṛtavān* 'Devadatta (has) made a mat'
(22) *devadattena kaṭo' kāri* 'A mat was made (has been made) by Devadatta'
(23) *devadattena kaṭaḥ kṛtaḥ* 'A mat was made (has been made) by Devadatta'.

---

[50] I use terms like 'inflectional morphology' and 'derivational morphology' merely as a convenient way to discuss data that western grammarians treat within these categories.

[51] A 2.3.50: *ṣaṣṭhī śeṣe* (PWT 166 [**251**]). How residual relations signified by sixth-triplet endings are connected with action-participant relations becomes the object of considerable discussion among Pāṇinīyas; see *Pāṇini and Pāṇinīyas on śeṣa relations* (*First Kunjunni Raja Memorial Lecture*), Aluva-Kerala: Kunjunni Raja Academy of Indological Research, 2008.

[52] A 2.3.19: *sahayukte' pradhāne* (PWT 164 [**249**]).

[53] One should not be misled by the way in which later Pāṇinīya works such as the *Siddhāntakaumudī* do indeed deal with nominal and verbal forms as though they were independent entities. Such treatments are intended to allow later students to learn Pāṇini's system and, despite their organization, they cannot help letting true Pāṇinian procedure show. For example, although the *Siddhāntakaumudī* describes first the formation of padas with nominal endings and only later considers rules such as A 2.3.2 (see note 39), it nevertheless has to bring in A 2.3.1 (see note 40) and to specify the affixes which are considered first to signify participants in actions (*kāraka*) and take precedence over nominal endings. It is not necessary to consider any details here.

(20) and (22) contain forms which derive from complexes with the L-affix ***luṅ***, introduced on condition that an agent is to be signified and the act in question is referred to the past:[54] *kṛ-l → kṛ-ti → kṛ-t → kṛ-s-t → kār-s-t → akār-s-t → akār-s-īt → akārṣīt, kṛ-l → kṛ-ta → kṛ-i-ta → akṛ-i-ta → akṛ-i-ø → akāri.*[55] (21) and (23) contain *kṛtavat-* (nom. sg. m. *kṛtavān*) and *kṛta-* (*kṛtaḥ*), derivates with the kṛt affixes ***ktavatu*** and ***kta***, respectively. In general, kṛt affixes signify agents, but ***kta*** is introduced to signify an immediate object of an action.[56] In addition, both these suffixes are introduced if an action is referred to the past.[57]

From such examples, it is clear that post-verbal affixes that are parts not only of finite verb forms but also of derived nominal bases are introduced as alternative expressions within syntactic derivations.

2.2.2.2 Taddhita Affixes. Affixes called *taddhita*, which modern grammars speak of as secondary derivate affixes, also form derived nominals that are based on the syntax. Most of these affixes are introduced optionally; and where a particular relation to another pada is involved, such an affix follows the first of the related padas referred to in the sūtra which introduces it.[58] For example, *mathurāvat-* and *devadattavat-* are derivates with the suffix ***vati*** following padas with seventh- and sixth-triplet endings. Such padas occur in strings like

(24) *mathurā-i iva-s srughna-i prākāra-s*
(25) *devadatta-as iva-s yajñadatta-as go-as*

which are posited as initial strings to account for the utterances

(26) *mathurāyām iva srughne prākāraḥ* 'The city wall in Srughna is like the one in Mathurā'
(27) *devadattasyeva yajñadattasya gāvaḥ* 'Yajñadatta's cows are like Devadatta's'

where *iva* 'like, similar' is used to bring out that *mathurā-i* and *devadatta-as* refer to entities spoken of in comparisons. What is expressed by (26)-(27) can alternatively be expressed by

(28) *mathurāvat srughne prākāraḥ*
(29) *devadattavad yajñadattasya gāvaḥ*

using *mathurāvat* and *devadattavat* instead of *mathurāyām iva* and *devadattasyeva.* Pāṇini accounts for this by allowing ***vati*** optionally to follow *mathurā-i* and *devadatta-as.* A rule provides that ***vati*** follows a pada that has a seventh- or sixth-

[54] A 3.2.110: *luṅ* (PWT 149 [**233**]).
[55] See PWT 157-58 (243), where the derivation of the comparable forms *apākṣīt* '(has) cooked' and *apāci* 'was cooked, has been cooked' are given.
[56] A 3.4.67, 70: *kartari kṛt | tayor eva kṛtyaktakhalarthāḥ* (PWT 161 [**248**]).
[57] A 3.2.102: *niṣṭhā* (PWT 198 [**294**]).
[58] A 4.1.82: *samarthānām prathamād vā* (PWT 69 [**111**]).
[59] A 5.1.116: *tatra tasyeva* (PWT 242 [**347**]). This and the preceding rule are objects of extensive discussion in the *Vākyapadīya* and elsewhere, but I cannot deal with the issues here.

triplet ending and is used in construction with *iva*;[59] the meaning conveyed by *iva* is then carried by the taddhita suffix.

Derivates with taddhita affixes are themselves derived nominal bases, and a rule provides for deleting a nominal ending included in such a base:[60] *mathurā-i-vat* → *mathurāvat, devadatta-as-vat* → *devadattavat*[61] Thus, starting with the initial strings (24) and (25), alternative paths are allowed: one can proceed with the operations that apply for the related padas, to derive (26) and (27), or one can let *mathurā-i*, *devadatta-as* take ***vati*** expressing the meaning of *iva-s*, and thereby derive *mathurāvat* and *devadattavat*, which then enter into construction with *srughna-i prākāra-s* and *yajñadatta-as go-as* to derive (28) and (29), equivalent to (26) and (27).

2.2.2.3 Compounds. Just as (26) and (28) (§2.2.2.2) are equivalent expressions, so are

(30) *rājñaḥ puruṣaḥ prāsāde ni vasati*
(31) *rājapuruṣaḥ prāsāde ni vasati* 'The king's servant stays in the palace.'

Moreover, Pāṇini no more considers analytic complexes like *rājñaḥ puruṣaḥ* and items like *rājapuruṣaḥ* independent entities unrelated to each other than he considers complexes such as *mathurāyām iva* and items like *mathurāvat* to be independent and not related with each other.[62] On the contrary, the related entities are accounted for by providing that a pada with a sixth-triplet ending, such as *rājan-as* in

(32) *rājan-as puruṣa-s prāsāda-i ni-s vas-l*

---

[60] A 2.4.71: *supo dhātuprātipadikayoḥ* (PWT 186, 229 [**278**, **330**]). The seventh-triplet ending ***ṅi*** of *mathurā-i-* is introduced to denote a locus, by A 2.3.36: *saptamy adhikaraṇe ca* (PWT 155-56 [**240**]). Under another interpretation, advanced notably by Nāgeśa, the ending has the value of a sixth-triplet ending. It is neither possible nor necessary to discuss this issue here.

[61] The derived nominal then itself takes a first-triplet ending ***su***, which is also deleted. The set of items beginning with *svar* and assigned to the avyaya class (A 1.1.37: *svarādinipātam avyayam*, PWT 27 [**54**]) includes derivates in *-vat*; nominal endings and feminine suffixes referred to by ***āp*** are deleted after avyayas (A 2.4.82: *avyayād āpsupaḥ*, PWT 212 [**307**]). *iva* is included in the set of items, beginning with *ca*, assigned to the class called *nipāta* (A 1.4.57: *cādayo' sattve*, PWT 28 [**55**]). By A 1.1.37, this too belongs to the avyaya group, so that an ending introduced after it is deleted.

[62] The possibility of treating taddhita derivates like *pañcāla-* 'Pañcāla district' (see (9), §1.2.3) as totally independent lexical items instead of terms derived from *pañcāla-* 'Pañcāla' and similarly considering compounds like *rājapuruṣa-* and complexes such as *rājñaḥ puruṣaḥ* as though they were completely independent of each other was seriously entertained. Thus, Kātyāyana brings up the possibility of doing away with the options provided in the *Aṣṭādhyāyī* for deriving nominals with taddhita suffixes and compounds from members of syntactic strings, and at least one grammar, the *Kātantra*, in fact did without rules of taddhita affixation and composition. For a discussion of these issues, see 'Theoretical precedents of the *Kātantra*', in *Grammatical traditions of Kashmir: Essays in honour of Pandit Dinanath Yaksha* (ed. Mrinal Kaul and Ashok N. Aklujkar), Delhi: D. K. Printworld, pp. 300-67. See also note 65.

optionally forms a compound with a related pada, such as *puruṣa-s* of the same string.[63] (30) and (31) are thus treated as alternative expressions starting with the initial string (32). If composition does not apply, (30) is derived by applying operations whereby the L-affix ***laṭ*** is replaced by the ending ***tip***, the suffix ***śap*** follows *vas-* of *vas-ti*, the ending of *ni-s* is dropped, the penultimate *-a-* of *rājan-* in *rājan-as* also is deleted, and phonological rules apply to form *rājñaḥ* and *puruṣaḥ* from *rājn-as* and *puruṣa-s*. If, alternatively, composition applies, *rājan-as* and *puruṣa-s* are combined to form the derived nominal base {*rājan-as-puruṣa-s*}. The endings included in this base are deleted (A 2.4.71, see note 60): *rājan-as-puruṣa-s* → *rājapuruṣa-*.

Now, *puruṣa-* in (32) is coreferential with the L-suffix of *vas-l* in this string: the latter signifies an agent, and the former refers to a particular person who plays this role in the act of staying. Since an agent has thus been signified by the verbal suffix, the nominal base takes a first-triplet ending (A 2.3.46, see note 9), and because there is a single agent, the singular endings ***tip*** and ***sup*** are selected (A1.4.22, see note 13). The compound *rājapuruṣa-* also refers to a man who bears a relation to a king and who plays the role of agent in the act of staying in the palace. Thus, *rājapuruṣa-* also is coreferential with the L-suffix of *vas-l*.

2.2.3 Number in derivates

2.2.3.1 Undifferentiated Number. One of the features that distinguishes compounds from corresponding analytic structures is the lack of number distinction in a constituent of a compound.[64] For example, (31) can be equivalent not only to (30) but also to

(33) *rājñoḥ puruṣaḥ prāsāde ni vasati*
(34) *rājñām puruṣaḥ prāsāde ni vasati.*

That is, *rājapuruṣaḥ* does not specify whether the servant in question is related to one king (gen. sg. *rājñaḥ*), two kings (gen. du. *rājñoḥ*) or several kings (gen. pl. *rājñām*).

The question arises: just how should one consider and account for these facts? In this context, let us consider two major points of view Bhartṛhari deals with, the second of which has two subtypes. First, (C), any number distinction (*saṅkhyābhedaḥ*) simply ceases to be (*nivartate*) in a constituent (*avayave*) of a derivate (*vṛttau*) such as a compound or taddhita formation. In a compound like *rājapuruṣa-*, then, the first constitutent (*rāja-* ← *rājan-*) merely resembles (*tadrūpe* 'of that form') the item that occurs in a constituent pada of an utterance like (30), (33), (34), from which it differs by not having any number distinction at all.[65] Alternatively, (D), there comes into play

[63] A 2.2.8: *ṣaṣṭhī* (PWT 206 [**301**]). The pada combines with another pada only if the two are syntactically and semantically related (*samartha*), as provided for by A 2.1.1: *samarthaḥ padavidhiḥ* (PWT 66, 206-7 [**109**, **301**]).

[64] This is one of a series of differences that set compounds apart from analytic strings, as given in a ślokavārttika cited and illustrated in the *Mahābhāṣya* on A 2.1.1 (I.362.13-363.6/II.509-12). Patañjali characterizes these differences as results of the meaning unification (*ekārthībhāvakṛtāḥ*) involved in the formation of compounds. For example, in *rājñaḥ puruṣaḥ*, the independently accented padas (*rājñaḥ purùṣaḥ*) refer separately to a man and the king to whom he bears a relation, but *rājapuruṣá-* is a distinct base with a single principal meaning, a man qualified by his relation to a king.

[65] This stand comes very close to the position under which compounds and analytic strings are treated as totally independent entities, so that a rule providing that compounding is optional

(*upajāyate* 'arises') for the subordinate meaning of a constituent in a derivate a number that is absolutely different (*anyaiva*) from the singularity, duality, and plurality that inheres in the significands of terms in strings like (30), (33), (34). This number (*saṅkhyā*) has the form of undifferentiated singularity (*abhedaikatva*- 'singularity characterized by nondifference'). That is, the first constituent of a compound such as *rāja-puruṣa*- or a taddhita derivate such as *pañcāla*- (§§1.2.3, 1.2.4) or *mathurāvat*- (§2.2.2.2) is considered to have a singular number that is distinct from the singularity which contrasts with duality or plurality. (D) has two versions.

First, (D1), the numerosity in question and called *abhedaikatvasaṅkhyā* takes the form of an undivided (*avibhaktam*) combination of numbers (*saṁsargarūpaṁ saṅkhyānām*).[66] As Helārāja explains, a subordinate significand (*upasarjanapadārthānām* 'of subordinate word meanings') of a term like *rāja*- in a derivate has a necessary connection with the quality of number since it is a thing (*sattvabhūtatvāt* 'because it has the status of a being'). However, in a derivate like *rājapuruṣa*-, a particular number is not determined. Hence, one infers that there is a general number which combines all particular numbers, which takes all these into its bosom, so to speak.[67] Bhartṛhari himself likens this to the combination of juices of medicinal plants

---

becomes unnecessary (see note 62). Position (C) is the first one which Bhartṛhari brings up in connection with his discussion of number and derivates, in VP 3.14.99: *vācikā dyotikā vāpi saṅkhyānāṁ yā vibhaktayaḥ | tadrūpe' vayave vṛttau saṅkhyābhedo nivartate* || At the end of his comments on this kārikā, Helārāja brings out the essential point as follows. The subordinated meaning of a constituent like *rāja*- in *rājapuruṣa*- is absorbed in the principal meaning of the whole compound, so that there is no distinction of independent meanings of constituents and any particular number associated with such a meaning ceases to be. Since this is absent, then, the nonoccurrence of particular endings is natural. Of course, the fact remains that Pāṇini provides for deleting nominal endings in compounds (A 2.4.71, note 60), which implies that they are introduced in the first place. Hence, it is proposed that the deletion of such endings is taught in order to allow operations which would be conditioned by affixes if they occurred (A 1.1.62: *pratyayalope pratyayalakṣaṇam*, PWT 63 [**103**]). Thus, *rājan*- in the compound is a pada, so that its *-n* is dropped (A 8.2.7: *nalopaḥ prātipadikāntasya*, PWT 347 [**539**]). Helārāja ends by noting the import of this view: a string like *rājñaḥ puruṣaḥ* and a derivate like *rājapuruṣa*- of *rājapuruṣaḥ* bear a mere similarity; there is no essential unity of the two—which, if conceived of, is an error due to the similarity—since in the string there is a number difference and this fails to occur in the derivate. VPH 3.14.99/98 (201.2-5): *prādhānārthasambhedād upsarjanārthasya bhedābhāvān nivartate saṅkhyāviśeṣa iti tadabhāvāt svābhāviky evānutpattir vibhaktīnām | pratyayalakṣaṇārthaṁ lug anvākhyāyate | itthaṁ sādṛśyamātraṁ bhrāntaṁ vṛttivākyayor na tattvata aikyaṁ vākye saṅkhyābhedasadbhāvād vṛttau tadabhāvād iti tātparyārthaḥ* | Thus, (C) fits in a scheme that is essentially non-Pāṇinian (cf. note 62). Accordingly, in his introduction to VP 3.14.100/99, Helārāja characterizes this as 'the view of some': *evaṁ tāvad upasarjanārtho nivṛttasaṅkhyābheda ity ekīyaṁ matam* | (VPH 201.6). Moreover, providing for deleting the *-n* of *rājan* in a compound presupposes that this is a pada, which goes against considering *rājapuruṣa* a totally independent lexical item.

66 VP 3.14.100: *abhedaikatvasaṅkhyā vā tatrānyaivopajāyate | saṁsargarūpaṁ saṅkhyānām avibhaktaṁ tad ucyate* ||

67 VPH 3.14.100/99 (201.9-11): *vṛttāv upasarjanapadārthānāṁ sattvabhūtatvād āvaśyakaḥ saṅkhyāyogaḥ | na hy avyayavan niḥsaṅkhya upasarjanārtho yuktaḥ sattvamātrasya sampratyayāt | na ca viśeṣo' vadhāryate | ataḥ sāmānyasaṁkhyārūpaṁ sarvasaṅkhyāviśeṣāṇāṁ saṁsargarūpaṁ kroḍīkārakaṁ vidyata ity anumīyate* |

blended in honey: they impart to honey their capacities, which they thereby give up as individuals; they all occur without division among them. The number in question is acknowledged to be of this sort.[68] Under D1, then, the three possible numbers that could inhere in the significands of terms such as *rājñaḥ, rājñoḥ, rājñām, pañcālānām* used in strings like (13) (§1.2.4), (30), (33), (34) come to occur undivided (*avibhāgena*) in derivates such as *rājapuruṣa-, pañcāla-*.[69]

While according to D1 singular, dual, and plural numbers are allowed to persist, though now combined in a single whole that is distinct qua a number of undifferentiated singularity (*abhedaikatvasaṅkhyā*), under D2 one abstracts from the individual numbers to a generic property (*jāti*) of being a number (*saṅkhyātva*) that inheres in any number. As a consequence of giving up the individual distinctions (*bhedānām ... parityāgāt*), due to the effect of the generic property having priority (*vyāpārāj jātibhāgasya* 'because of the generic part's action'), the undifferentiated entity referred to as *abhedaikatvasaṅkhyā* now occurs characterized by the removal of individuals (*bhedāpohena vartate*).[70] Helārāja notes how the singularity of nondifferentiation (*abhedaikatvam*) as now conceived can be considered a number (*saṅkhyātvam* 'being a number'): as singularity excludes duality, and the latter excludes threeness and so on, similarly this excludes all individual singularity and so on, so that it has the property of being a number.[71]

In sum, under D, the meaning of terms such as the first constituents in the compound *rāja-puruṣa-* and so on are considered not to be characterized by any specific contrastive number. They are treated as being loci of singular, dual or plural number now lumped together as an undifferentiated whole (D1) or of any indeterminate number, characterized by the property of being a number (D2).

2.2.3.2 Number Distinctions and Derivation. An utterance such as (20) *devadattaḥ kaṭam akārṣīt* (§2.2.2.1) is accounted for in Pāṇini's system as derived from an initial string

(35) *devadatta-s kaṭa-am kṛ-l.*

The pada *kaṭa-am* in this string has a specific nominal ending: *am*, the ekavacana member of the first triplet of endings. Now, A 2.3.2 (see note 39) does not apply in

---

[68] VP 3.14.101: *yathauṣadhirasāḥ sarve madhuny āhitaśaktayaḥ | avibhāgena vartante tāṁ saṅkhyāṁ tādṛśīṁ viduḥ* ||

[69] In his *Vaiyākaraṇabhūṣaṇa* and its abridgement, Kauṇḍabhaṭṭa says that what is called *abhedaikatvasaṅkhyā* is particular numbers (*saṅkhyāviśeṣāṇām*) residing (*avasthānam*) without division (*avibhāgena*), their being (*sattvam*) without division: VBh. 56 (272): *saṅkhyāviśeṣāṇām avibhāgenāvasthānam abhedaikatvasaṅkhyā*, VBhs. 56 (417): *saṅkhyāviśeṣāṇām avibhāgena sattvam abhedaikatvasaṅkhyā*. In both places, he goes on to cite VP 3.14.101. In his comments on Bh. 2.1.1, Kaiyaṭa (Pr. II.509/II.323b) gives the first definition, then goes on to cite VP 3.14.101-103.

[70] VP 3.14.102: *bhedānāṁ vā parityāgāt saṅkhyātmā sa tathāvidhaḥ | vyāpārāj jātibhāgasya bhedāpohena vartate* || I abstain from dealing with small, though important, differences of interpretation that arise from different ways of considering the terms *saṅkhyātmā* and *tathāvidhaḥ*.

[71] VPH 3.14.102/101 (202.10-11): *yathā hy ekatvaṁ dvitvādy apāvartayati dvitvaṁ ca tritvādy evam idam abhedaikatvaṁ sarvam ekatvādy apāvartayatīty asti saṅkhyātvam* |

isolation to introduce an undifferentiated triplet of endings {*am auṭ śas*} after a nominal. It applies in conjunction with A 1.4.21-22 (see note 33), so that a specific ending is selected immediately on the basis of the number in question, which is a semantic property that serves as one of the conditions for affixation.

Thus, although position (D1) can be countenanced if one considers the possible ambiguity of a derivate like *rājapuruṣa-*, once Pāṇini's derivational system is brought into play, a difficulty arises: where in the derivational history is there an undifferentiated sixth triplet of endings {***ṅ****as os ām*} that can be associated with the undifferentiated number? Further, one may infer a generic property such as the property of being a cow or ox (*gotva*) from experience. Such an abstract property accounts for the fact that each time someone sees a cow or ox, that person experiences the cognition of its sameness (*abhinnapratyaya* 'undifferentiated cognition'): a cow or ox is seen every time. A common generic property (*sāmānyam*) is thus justified by identical cognitions (*abhinnapratyayāvedyam* 'made known by ...'). There is no comparable effect which one can attribute to the undifferentiated singularity at issue, since only individual contrastive numbers are associated with operations provided for in rules of the grammar. Hence one can justifiably ask what inferential mark (*liṅgam*) there is which points to its existence (*tatsadbhāve*).

Helārāja poses this problem[72] by way of introducing two kārikās[73] where Bhartṛhari confronts the issue, as follows. Because it is a substance, a given object seen indistinctly from afar is perceived as endowed with some color, but it is not grasped as having a particular color such as white. The situation is comparable if the inherence of a specific number is not intended,[74] as in the first constituent of *rāja-puruṣa-*. Further, one's understanding that the referent of such a constituent has some number (*saṅkhyāvattvasampratyayaḥ*) itself now is the effect accounted for by the assumed undifferentiated singularity (*abhedaikatvasaṅkhyākāryam*).[75]

In terms of a derivational system, (D) could be accommodated by assuming that the derived nominal base *rāja-puruṣa-* is formed from a posited complex like

(36) *rājan-as puruṣa-s*

in which the the ekavacana ending ***ṅ****as* in *rājan-as* does not designate a relation with respect to a single king as opposed to two or more kings; instead, there is reference to any of these or to a numerosity in general. Under (D)—especially under (D2)—this amounts to treating a singular ending attached to a base that enters into composition as a generic singular, introduced when one speaks of a group in general, characterized

---

[72] VPH 3.14.103-104/102-103 introduction (202.13-14): *nanu cābhinnapratyayāvedyaṁ sāmānyaṁ gotvādi | atra cābhedaikatvasya na kiñcit kāryam | bhedā eva hi śāstrakāryayogina iti kiṁ tatsadbhāve liṅgam ity āśaṅkyāha |*

[73] VP 3.14.103-104: *agṛhītaviśeṣeṇa yathā rūpeṇa rūpavān | prakhyāyate na śuklādibhedarūpas tu gṛhyate || bhedarūpasamāveśe tathā saty avivakṣite | bhāgaḥ prakāśitaḥ kaścic chāstre' ṅgatvena gṛhyate ||*

[74] *bhedarūpasamāveśe tathā saty avivakṣite.* Helārāja (202.20) glosses this with *saṅkhyāviśeṣasamavāye pratipādyatvenāniṣṭe* 'when the inherence of a particular number is not intended as what is to be conveyed'.

[75] VPH 3.14.104/103 (202.22-23): *tathā ca saṅkhyāvattvasampratyaya evātropasarjanārthasyābhedaikatvasaṅkhyākāryam |*

by a generic property. Such a singular alternates with a plural.[76] This brings us back full circle to position (C) in that it is now possible to consider that the constituent of a derivate such as *rājapuruṣa-* is any king as characterized by the property of being a king.[77]

Bhartṛhari himself confronts the issue of plural forms being used in compounds such as *goṣucara-* 'one that goes around cattle'.[78] In the second half of VP 3.14.107

[76] For example, one can say *sampannā vrīhayaḥ* or *sampanno vrīhiḥ* ('The rice is rich, full grown'), *brāhmaṇāḥ pūjyāḥ* or *brāhmaṇaḥ pūjyaḥ* ('A brāhmaṇa is to be honored' = 'Brāhmaṇas are to be honored'). Such usage is accounted for by A 1.2.58: *jātyākhyāyām ekasmin bahuvacanam anyatarasyām*, according to which a bahuvacana ending optionally occurs with reference to a single entity, when one conveys a generic meaning (*jātyākhyā-yām*).

[77] Nāgeśa brings this out in his *Uddyota* on Pr. 2.1.1 (Ud. II.511/II.325a), when he states a view others (*pare tu*) hold—that is, which he prefers—namely that in a derivate one or more individuals appear as referents, in accordance with a speaker's intent, solely as qualified by a property such as being a king, not by properties of numerosity, being three, four and so on: ... *tathā vṛttau rājatvādinaivaikānekavyaktibhānaṁ tātparyavaśān na tu kadācid api tritvacatuṣṭvādinā* | He ends his statements on this particular issue by noting pertinently that nowhere in the Bhāṣya is the statement found that the singular number of non-differentiation appears (*abhedaikatvasaṅkhyā bhāsata iti tu bhāṣye kvāpi na dṛśyate*). Note, however, that Bhartṛhari himself appears to consider that position (D) was implicitly accepted by Kātyāyana. For he says (VP 3.14.107: *alukaś caikavadbhāvas tasmin sati na śiṣyate* | *sa ca goṣucarādīnāṁ dharmo'sti vacanāntare* ||) that, under the assumption that undifferentiated singularity applies (*tasmin sati*) in compounds, treating the meaning of a subordinate constituent as singular when an ending is not deleted (*alukaḥ ... ekavadbhāvaḥ*) is not explicitly taught (*na śiṣyate*). Helārāja introduces this kārikā noting that Bhartṛhari here speaks of something which shows that Kātyāyana agrees (VPH 3.14.106/107 [203.14]: *atra vārttikakārīyaṁ saṁvādakam āha*). In his commentary on the kārikā, Helārāja then alludes (203.17-204.3) to what Kātyāyana and Patañjali say, in the discussion of A 6.3.1: *alug uttarapade* (PWT 224 [**323**]), concerning A 6.3.2: *pañcamyāḥ stokādibhyaḥ*. This sūtra provides for not deleting a fifth-triplet ending after a set of terms, starting with *stoka-* 'little', as in *stokānmukta-* 'freed with little effort'. Patañjali (Bh. III.141.13-14/IV.582: *ekavac cālug bhavatīti vaktavyam* | *kiṁ prayojanam* | *stokābhyāṁ muktaḥ stokebhyo mukta iti vigṛhya stokān mukta ity eva yathā syāt* |) introduces the discussion by proposing that one should say explicitly that where an ending is not deleted the meaning is treated as singular. The reason given for this is so that the compound *stokānmukta-* alone be allowed, even if a compound *stokānmukta-* is considered to correspond to analytic sequences *stokābhyāṁ muktaḥ, stokebhyo muktaḥ*, with dual and plural forms. In his second and third vārttikas (6.3.1 vt. 2-3: *ekavadvacanam anarthakam* | *dvibahuṣv asamāsaḥ* |), Kātyāyana retorts: stating that nondeletion is treated as singular serves no purpose, since a compound is not formed in the first place from padas with dual and plural endings. Moreover, he goes on to say (6.3.1 vt. 4: *uktaṁ vā* |), the reason for this has been stated elsewhere (3.2.1 vt. 5: *anabhidhānāt* |): a compound is not formed because it is not used to express the meanings of analytic sequences with dual and plural forms. Now, Helārāja cites vārttikas 2-3 but not vt. 4, so that he can indeed claim Kātyāyana's agreement with the thesis under discussion, and he does so because Bhartṛhari himself assumes this. Although it is impossible to take up the issue here in detail, I think, nevertheless, that Kātyāyana does not in fact give evidence of operating with position (D).

[78] A 3.2.16: *careṣ ṭaḥ* (*adhikaraṇe* 15) introduces *ṭa* after *car* 'move, go about' construed with a cooccurring nominal that denotes a locus. A derivate with this kṛt suffix obligatorily

(see note 77), he remarks that this property—that is, singularity characterized by nondifferentiation—pertains to the referents of first constituents in compounds like *goṣucara-*, where an affix associated with a number other than singular is involved. Commenting on this, Helārāja[79] remarks that this half verse answers to the possible question: if a number associated with the meaning of a compound constituent is that of undifferentiated singularity, how is it that some compounds include constituents with plural endings, as in examples like *goṣucara-, varṣāsuja-* ('born in the rainy season'), *apsuyoni-* ('whose source is in water')? The property of singularity characterized by nondifferentiation is to be found also in such instances, where there is an ending other than a singular ending. For in these examples the plural ending does not refer to plurality of individuals (*vyaktibahutve*). The plural is accounted for as an alternative to a generic singular, by A 1.2.58 (see above with note 77). Accordingly, one understands here also only singularity associated with lack of differentiation. Thus, a creature that goes about a single cow also is referred to as *goṣucara*;[80] even an atom of water is referred to by *apsu-*, with the plural ending; even a single night of the rainy season is referred to by the term *varṣā* with a plural ending. All this is due to the very nature of how words signify (*śabdaśaktisvābhāvyāt* 'due to the nature of the capacities of words'), so that assuming a singular of nondifferentiation as a number characteristic of items in derivates does not preclude differences in endings in such constituents.

There are other instances, however, where Bhartṛhari admits that even a compound constituent has indeed to be associated with a specific contrastive number. He considers in particular the derivates *śaurpika-* 'bought with a śūrpa (of grain)', *māsajata-* ('born a month earlier'). The first is derived by introducing the taddhita suffix ***ṭhañ***

---

forms a compound with a pada denoting the locus in question (A 2.2.19: *upapadam atiṅ*, PWT 219 [**315**]). In addition, as provided for in A 6.3.14*: tatpuruṣe kṛti bahulam* (*alug uttarapade* 1), the ending of a first-constituent pada is variously exempted from deletion before a second constituent with a kṛt affix. *goṣucara-* means 'rooster' (*kukkuṭa*) according to Kaiyaṭa (Pr. II.510/II.324b), who notes that a rooster is spoken of without distinction of whether it lingers around one, two or many heads of cattle: *goṣucaraḥ kukkuṭa ucyate* | *ekasyāṁ gavi dvayor bahuṣu vā yaś carati sa sarvo' sāv aviśeṣeṇocyate* | *goṣucara-* is one of three compounds Patañjali cites in the Bhāṣya on A 2.1.1 while arguing that a speech form should depend on a meaning and not vice versa (see note 49): if one maintained that a meaning depends on a speech form, then one would have to accept that a number difference is understood in *apsucara-* ('one who goes about in the water'), *goṣucara-* and *varṣāsucara-* ('one that goes about during the rainy season'). Kaiyaṭa also glosses *varṣāsucara-* as *indragopa-*, which refers to a particular insect (cf. HemAbh. 4.275cd: *indragopas tv agnirajo vairāṭas tittibho 'gnikaḥ* ||).

[79] VPH 3.14.107/106 (204.3-10): *kathaṁ tarhi goṣucaro varṣāsujo'psuyonir iti bahuvacanānāṁ samāsa iti ced āha: sa ceti* | *so' bhedaikatvākhyo dharmo vacanāntare ekavacanād anyasmin bahuvacane'py atrāsti* | *atra hi nedaṁ vyaktibahutve bahuvacanam api tu jātyākhyāyām ekasmin bahuvacanam iti* | *tathā caikasyām api yo gavi carati prāṇī so'pi goṣucara indragopo'bhidhīyate* | *eko'py cāpparamāṇur apchabdena bahuvacanāntenābhidhīyate / varṣārātro'pi cābhinno varṣāśabdena bahuvacanāntena śabdaśaktisvābhāvyād ity atrābhedaikatvasaṅkhyā vacanabhedapratibandhaṁ na karoti* | *tatrāpy abhedaikatvasyaiva pratīter ity arthaḥ* |

[80] Helārāja identifies the animal as the bug called *indugopa*. Others consider that *goṣucara* refers to a rooster (see note 78).

after a pada with a third-triplet ending to form a nominal base meaning 'bought with ...':[81]

(37) *śūrpa-ā-ṭha*

in which the taddhita suffix *ṭhañ* has the meaning 'bought' (see note 81). *māsajāta-* is a tatpuruṣa compound derived by combining a pada consisting of a time word (*kāla*) followed by a first-triplet ending with a pada in which a sixth-triplet ending follows a base denoting someone whose age is measured (*parimāṇinā* 'who has a measure') by the time in question:[82]

(38) *māsa-s jāta-as.*

Now, although (38) is comparable structurally to (36), it is not appropriate to operate here with an indeterminate number for the referent of *māsa-*, and the same holds for the referent of *śūrpa-* in (37). For, the referent of each of these is a contrastive measure (*parimāṇam*): a śūrpa as a measure of the amount of grain with which something is bought and a month as the length of time that has elapsed after someone's birth. As can be seen, each measure serves as a delimitor (*paricchedaḥ*): something is bought with one śūrpa as opposed to two or more śūrpas; a single month has elapsed since someone was born. This limitation with respect to someone of a certain age (*vayasvini*) or something that has been bought (*krīte*), as signified by *māsajāta-* and *śaurpika-*, is desired and accepted (*iṣṭaḥ*), but cannot be understood (*na gamyate* 'is not understood') unless a number difference is accepted (*bhedād ṛte* 'without a difference'); without this, a defining measure (*parimāṇam*) is purposeless (*anarthakam*). Hence, Bhartṛhari accepts that in such instances a measure applies distinctively (*bhedena vartate*)—that is, in opposition to measures delimiting other entities—and does so of its very nature (*svabhāvataḥ*), on the basis of a number (*āśritya saṅkhyām*) which serves as a qualifier (*upādhibhūtām*) of the referents of *śūrpa-* and *māsa-*.[83]

Accordingly, after presenting position (D2) and supporting it with a parallel,[84] Bhartṛhari goes on (VP 3.14.104cd) to remark that in particular instances a part manifested in some way (*bhāgaḥ prakāśitaḥ*) is accepted (*gṛhyate* 'is taken, understood') as a condition (*aṅgatvena*) in a rule (*śāstre*). That is, a distinctive number is at times understood as a condition for a derivational rule applying, even though an ending associated with a specific number does not appear in a derivate.

Another example of a rule that takes a specific number into consideration in accounting for derivates involves *tāvaka-* 'pertaining to you, your', *māmaka-* 'pertaining to me, my', equivalent to *tava* 'your' and *mama* 'my'. These alternate with other equivalents of genitive pronominal forms: *tāvakīna-, māmakīna-* and *tvadīya-, madīya-*.

---

[81] A 5.1.37: *tena krītam* (PWT 238 [**342**]). A 5.1.26: *śūrpād añ anyatarasyām* provides that ***añ*** optionally follows a pada whose base is *śūrpa-*; by A 5.1.18: *prāg vateṣ ṭhañ* (PWT 235 [**338**]) the suffix *ṭhañ* also applies, so that *śaurpika-* (← *sūrpa-ā-ṭha*) and *śaurpa-* (← *śūrpa-ā-a*) are derived as alternates. In both derivates, *śūrpa-* denotes a measure (*parimāṇa*) of grain or such.

[82] A 2.2.5: *kālāḥ parimāṇinā*. For a discussion of this rule, see JAOS 111 (1991): 447-48.

[83] VP 3.14.126-127: *śaurpike māsajāte ca parimāṇaṁ svabhāvataḥ | upādhibhūtām āśritya saṅkhyāṁ bhedena vartate || vayasvini paricchedaḥ krīte cāpi na gamyate | iṣṭo bhedād ṛte tatra parimāṇam anarthakam ||*

[84] VP 3.14.103-104; see above with notes 73-74.

They also contrast with derivates equivalent to genitive dual and plural forms: *yauṣmāka-, yauṣmākīna-, yuṣmadīya-* (equivalent to *yuvayoḥ* [du.], *yuṣmākam* [pl.]), *āsmāka-, āsmākīna-, asmadīya-* (equivalent to *āvayoḥ, asmākam*). Pāṇini accounts for these by providing that padas consisting of the pronominal bases *yuṣmad* and *asmad* and sixth-triplet endings are followed by *khañ* as well as *cha* and, optionally, *aṇ*.[85] In addition, before the affix *khañ* (*tasmin* 'before it') as well as *aṇ*, these pronominal bases are respectively replaced by *yuṣmāka-* and *asmāka-*; but, if they are singular (*ekavacane* 'signifying one') they are replaced by *tava-* and *mama-*.[86] In accordance with such usage, Bhartṛhari accepts[87] that a particular number which inheres in the meaning of a base[88] is manifested (*vyajyate*), even without an ending (*vibhaktyā vinā*), through replacements (*ādeśaiḥ*).[89]

There are thus facts requiring an adherent of position (D) to accept that distinctions of contrastive number are not always eliminated in derivates.[90] Accordingly, as Helārāja notes,[91] it is not possible to state that a compound is not formed from padas with dual and plural endings, since composition does apply in such cases where a particular meaning is understood for a constituent of a compound. In examples like *rājapuruṣa-*, on the other hand, Helārāja goes on to say, there is no cause for understanding such a particular meaning (*viśeṣāvagamananibandhanābhāvāt*). Further, in the derivational procedure (*prakriyāyām*) intended to explain a derivate, a string (*vākyasya*) with items that are similar to those in a derivate is posited (*kalpanāt* 'due to a

---

[85] A 4.3.1: *yuṣmadasmador anyatarasyāṁ khañ ca.* The taddhita suffix applies by the major heading A 4.1.83: *prāg dīvyato'ṇ* (PWT 232 [**335**]). By A 1.1.74: *tyadādīni ca* (PWT 34 [**61**]) *yuṣmad* and *asmad* are assigned to the vṛddha class, so that *cha* follows a pada with one of these bases (A 4.2.114: *vṛddhāc chaḥ*, PWT 239 [**343**]). Affix-initial *kh, ch* are replaced by *īn, īy* (see PWT 331 [**509**]).

[86] A 4.3.2-3: *tasminnaṇi ca yuṣmākāsmākau, tavakamamakāv ekavacane.*

[87] VP 3.14.125: *upādhibhūtā yā saṅkhyā prakṛtau samavasthitā* | *ādeśaiḥ sañjñayā vāpi vibhaktyā vyajyate vinā* ||

[88] *yā saṅkhyā prakṛtau samavasthitā.* Although literally this speaks of a number which is located (*samavasthitā*) in a base (*prakṛtau*), it is appropriately interpreted to refer to the contrastive number (*bhedasaṅkhyā*) that inheres (*samavetā*) as delimiting a base's meaning (*tadarthasyāvacchedakatvena*). Thus Helārāja (VPH 3.14 125/124 [211.22-23]): ... *yā prakṛtiḥ prātipadikaṁ tadviṣaye tadarthasyāvacchedakatvena yā bhedasaṅkhyā samavetā* ....

[89] In VP 3.14.125 (note 87), Bhartṛhari says that a contrastive number is manifested not only by replacements but also by a class name (*sañjñayā*). This concerns terms like *kumārī-* in compounds, which under a particular interpretation of a Pāṇinian sūtra can be considered to have the class name *pragṛhya*. I do not take this up because a discussion would have to be fairly long and also because the interpretation under which the argument is advanced by Kātyāyana is part of a pūrvapakṣa; see note 92 for a brief presentation of the issue.

[90] In various places (VPH 3.14.100/99 [201.14], 102/101 [202.11], 103-104/102-103 [202.21], 125/124 [212.8-9]), Helārāja refers to these as conditions (*nimitta, nibandhana*) due to which contrastive numbers are understood in derivates.

[91] VPH 3.14.125/124 (212.7-12): *itthaṁ ca kṛtvā dvivacanabahuvacanāntānām asamāsa iti nedaṁ vacanaṁ sambhavati viśeṣāvagame samāsapravṛtteḥ* | *rājapuruṣādau tu viśeṣāvagamanibandhanābhāvād vṛttipadam anvākhyātuṁ prakriyāyāṁ sadṛśasya vākyasya kalpanād ekavacanāntānām eva tad yuktam iti nyāyasiddhārthānuvādo'yam atra* | *yatra tu dvitvādyabhivyaktisambhavas tatra samāsaprakṛtau prakriyāvākye kumāryor agāram ity ādāv upādīyata eva dvivacanādīti na kaścid virodhaḥ* |

construct'), so that it is appropriate that the posited string have only padas with singular endings, as in (36), since the derivate shows no number contrast in the first constituent. This is nothing more than a reiteration of what obtains by the nature of things. Where, however, there is a possibility of duality and plurality being manifested in a derivate, a dual or plural ending is indeed accepted in the string that serves as a source of a derivate. For example, the string

(39) *kumārī-os agāra-s*

posited as the source of the compound *kumāryagāra-* (← *kumārī-agāra-*) 'house of two girls' has a dual ending in *kumārī-os*.[92]

2.2.3.3 Conclusions. As the compound *māsajāta-* is derived from (38) *māsa-s jāta-as* (§2.2.3.2) in accordance with the semantics of the derivate, so also must *mathurāvat-* be derived from *mathurā-i*, with a singular ending (see (24), §2.2.2.2), since there is a single city of Mathurā. It is important to keep in mind, however, that (26) and (28) are to be accounted for as equivalent expressions. Pāṇini accounts for this by deriving both utterances as alternative results starting from a single abstract string (24).

By the same token, not only is *rājapuruṣaḥ* to be considered an equivalent of *rājñaḥ puruṣaḥ*, as in (31) and (30) (§2.2.2.3), in Pāṇini's system the compound is accordingly derived by a rule (A 2.2.8, note 63) which provides that a pada with a sixth-triplet ending optionally forms a compound with a related pada. Thus, if a compound *rājapuruṣa-* is to be derived, the padas in question must be syntactically and semantically related (*samartha*, see note 63), which means that the sixth-triplet ending introduced after *rājan-* must signify a relation that holds between a king and his servant(s) who reside(s) in the palace. Moreover, as I pointed out earlier (§2.2.3.2), a rule introducing a nominal ending does not serve to allow indiscriminately any member of a triplet of endings; instead, it applies in conjunction with rules specifying the choice of a particular ending associated with a particular number. The sole justifiable conclusion to be drawn from all this is that *rājan-as* in (32), where it is related to *puruṣa-s*—which itself refers specifically to a person that is agent of residing in the palace—serves to form only the first constituent of a compound *rājapuruṣa-* as in *rājapuruṣaḥ* of (31). In order to account for utterances that are homophonous with (31) and alternate with (33) and (34), these will have now to be derived from distinct strings in which *rājan-* is followed by a dual or a plural ending:

(40) *rājan-os puruṣa-s prāsāda-i ni-s vas-l*
(41) *rājan-ām puruṣa-s prāsāda-i ni-s vas-l.*

The same reasoning holds for derivates such as *tatra*, as in

(42) *yadi tv āryaṁ na śakṣyāmi vinivartayituṁ vanāt | vane tatraiva vatsyāmi yathāryo lakṣmaṇas tathā* 'If I can't get the noble (Rāma) to leave the forest, I will stay in that very forest, as will the noble Lakṣmaṇa'[93]

[92] Under a particular interpretation of A 1.1.11: *īdūded dvivacanam pragṛhyam* (PWT 43 [**76**]), *kumārī-* in *kumārī-agāra-* could be considered to have a pragṛhya segment, which requires that the constituent have a dual ending. I do not think it necessary to consider more details here.

where *tatra* is coreferential with *vane*. In Pāṇini's system, *tatra* of an utterance such as

(43) *tatra vane vatsyāmi* 'I will stay in that forest'

is derived from *tad-i*, as in

(44) *tad-i vana-i vas-l*

where the pronominal base *tad* 'that' and the nominal *vana* 'forest' are followed by the singular ending ***ṅi*** of the seventh triplet, introduced to signify a locus (see note 60), and *vas* 'dwell' is followed by the L-affix ***lṛṭ***, which signifies an agent and is introduced on condition that the action in question is referred to the future.[94] Starting from this string, two options are available. First, affixation and replacement rules can apply to give

(45) *tasmin vane vatsyāmi*

with the padas *tasmin,*[95] *vane,*[96] and *vatsyāmi.*[97] Optionally, the taddhita affix ***tral*** is introduced after *tad-i*[98] to form a derived nominal base *tad-i-tra*, the ending of which is deleted (see note 60): *tad-i-tra ... → tatra.*[99] This nominal base is coreferential with *vane* in (43), as is *tasmin* in (45). As a nominal base, *tatra* has to be followed by a nominal ending. Since, however, *tatra* already signifies a locus, a seventh-triplet ending is not introduced after this to signify such a kāraka; instead, the first-triplet ending ***su*** follows the base (see note 9). Further, *tatra* is assigned to the avyaya class,[100] so that its ending is dropped (see note 60): *tatra-s → tatra*. In this manner, (43) is accounted for as an alternative to (45).[101]

Now, *tatra* can be used with reference to any number of loci, not just one. Thus, in addition to (43), one can say

(46) *tatra vanayor vatsyāmi*
(47) *tatra vaneṣu vatsyāmi*

equivalent to

(48) *tayor vanayor vatsyāmi*

---

[93] *Rāmāyaṇa* 2.76.17, *The Ayodhyākāṇḍa, The Second Book of the Vālmīki Rāmāyaṇa ...*, critically edited by Dr. P. L. Vaidya, Baroda: Oriental Institute, 1962.

[94] A 3.3.13: *lṛṭ śeṣe ca* (PWT 149-50 [**233**]).

[95] *tad-i → taa-i* (A 7.2.102: *tyadādīnām aḥ*) → *ta-i* (A 6.1.97: *ato guṇe*) → *tasmin* (A 7.1.15: *ṅasiṅyoḥ smātsminau*); see PWT 313, 343, 322 [**476**, **531**, **492**].

[96] *vana-i → vane* (A 6.1.87: *ād guṇaḥ*, PWT 341-42 [**528**]).

[97] I do not consider it necessary to go through all the operations whereby *vatsyāmi* is derived from *vas-l*.

[98] A 5.3.10: *saptamyās tral* (PWT 244-45 [**349**]).

[99] *tad-tra → taa-tra → tatra* (see note 95). *tra* is classed as a vibhakti (A 5.3.1: *prāg diśo vibhaktiḥ*, PWT 40-41 [**71**]), hence conditions the replacement of *-d* by *-a* which also takes place before nominal endings, which too are assigned the class name *vibhakti* (A 1.4.103: *supaḥ*, PWT 39 [**67**]).

[100] A 1.1.38: *taddhitaś cāsarvavibhaktiḥ* (PWT 27 [**54**]).

[101] In fact, one could consider that *vane tatraiva* is used in (42) because of metrical requirements: *vane tasminneva* would make the pāda hypermetric.

(49) *teṣu vaneṣu vatsyāmi*

speaking of two (*vanayoḥ* [loc. du.]) and many (*vaneṣu* [loc. pl.]) forests. Nevertheless, it is not licit to say that all three utterances (43) and (46)-(47) can be derived in the Pāṇinian scheme from the same initial string (44). In that string, *tad* is coreferential (*samānādhikaraṇa*) with *vana*, both referring to a forest that plays the role of a place where someone will stay, so that it is assigned to the adhikaraṇa class.[102] A 2.3.36 (see note 60) therefore applies to introduce a seventh-triplet ending on condition that an adhikaraṇa is to be signified. Further, this sūtra again conforms to others of its kind in applying conjointly with rules that specify particular endings to be used when different numbers are involved. Since ***tral*** is introduced after a pada with a seventh-triplet ending, then, it has to occur after a term with a particular member of this triplet. Accordingly, just as (31) equivalent to (30) and (33)-(34) is appropriately derived from three distinct strings (32) and (40)-(41), so does Pāṇini's derivational procedure require positing not only (44) as the source of the alternants (43) and (45) but also

(50) *tad-os vana-os vas-l*
(51) *tad-su vana-su vas-l*

as distinct sources for the alternants (46), (48) and (47), (49).

That Pāṇini should thus discriminate among different sources for a single compound whose first constituent is ambiguous with respect to number accords with the manner in which he proceeds in other comparable cases. Now, as is well known, instrumental, dative, and ablative dual forms are homophonous, with an ending *-bhyām*. In western grammars, it is commonplace to see paradigms with one row labelled 'I. D. Ab.' for a single dual form in *-bhyām*. This is acceptable if one considers paradigms of forms and describes their syntactic uses. In Pāṇini's system, however, triplets of nominal endings are introduced under conditions (see §2.2.1) that require operating with distinct triplets: third triplet ***ṭ****ā bhyām bhis*, fourth triplet ***ṅ****e bhyām bhyas*, and fifth triplet ***ṅ****asi bhyām bhyas*. Though final forms may be homophonous, they differ in their origins according to the conditions under which the triplets to which they belong are allowed to occur.[103]

2.2.4 Gender Affixes in Derivation

As shown above (§2.2.3.3), *tad* and *vana* in (44) are coreferential bases. Similarly, in

(52) *tad-i aṭavī-i vas-l*

---

[102]A 1.4.45: *ādhāro' dhikaraṇam* (PWT 137-38 [**214**]).

[103]In the tradition of Pāṇinīyas and others, two positions are noted in this connection: single linguistic units are polysemous (*anekārthāḥ śabdāḥ*) or a difference of form is accepted due to a difference in meaning (*arthabhedāc chabdabhedaḥ*), so that one operates with homophonous distinct units relative to different meanings. Pāṇini adheres to the former position where appropriate. For example, instead of having separate verbal bases with distinct meanings, he operates with a single polysemous base such as *kṛ* ('do, make, snitch' etc.). The choice is dictated by the operations associated with items. In the present context, I cannot enter further into this issue.

tad of *tad-i* is coreferential with *aṭavī* ('forest') of *aṭavī-i*. In accordance with the Pāṇinian system, (52) is the source of two alternative expressions:

(53) *tasyām aṭavyāṁ vatsyāmi* 'I will stay in that forest'
(54) *tatrāṭavyāṁ vatsyāmi* 'I will stay in that forest.'

These utterances differ from (43) and (45) in that a forest is now referred to by a feminine (*aṭavī*) instead of a neuter (*vana*) term.

This has formal consequences. To begin with, the ending ***ṅi*** is replaced by *ām* after *aṭavī-*.[104] Now, in the Pāṇinian scheme not only is number a meaning that conditions operations but so also is gender. The stem *aṭavī-* of *aṭavī-i* is associated with this meaning from the outset. Further, *tad* used coreferentially with *aṭavī* has the same property. Accordingly, a feminine affix can be introduced. However, this is actually brought in only at a certain stage of derivation. First, before a nominal ending, the *-d* of a stem *tad-* is replaced by *-a*: *tad-i* → *taa-i* → *ta-i* (see note 95). At this point, there is a stem in *-a* which has been marked with femininity as well as singularity from the beginning of the derivation. Consequently, the feminine suffix ***ṭāp*** is introduced after this stem:[105] *ta-ām* → *ta-ā-ām* → *tā-ām*.[106] Two simultaneous operations now apply: the ending *-ām* following the pronominal stem that ends in the feminine suffix *-ā* receives the initial augment *syāṭ*, and the *-ā* of the stem is replaced by a short vowel:[107] *tā-ām* → *ta-syāām* → *tasyām*.

The derivate *tatra* formed from *tad-i* of (52) by introducing the affix ***tral*** also is coreferential with *aṭavī* in (54), so that the referent of this term too is characterized as feminine. Accordingly, the rule which introduces a feminine suffix ***ṭāp*** after a base in *-a* (see note 105) must be allowed to apply. Hence, Pāṇini provides that after a member of the avyaya class not only nominal endings but also feminine suffixes referred to by the abbreviation ***āp*** are absent (see note 61). In addition, since *tad-* in (52) refers to something qualified as feminine, once one reaches the state *ta-* in the derivation of *tatra* (see §2.2.3.3 with note 99), the condition is met for introducing the feminine suffix ***ṭāp*** after *ta-* of *ta-tra*. Pāṇini acknowledges this. For, he makes special provision that before a subset of taddhita suffixes—starting with ***tasil*** (e.g., *tatas* 'from that, from those, thence') and including ***tral***—bases are treated as masculine, so that a feminine affix does not occur after them.[108]

It is clear, then, that even certain feminine affix rules, which one might consider typically lexical in nature, are applied not in a lexicon apart from the syntax but within the syntactic derivational process itself.

---

[104] A 7.3.116: *ṅer ām nadyāmnībhyaḥ* (PWT 324 [**497**]). *aṭavī-ām* → *aṭavyyām* (A 6.1.77: *iko yaṇ aci*, PWT 78 [**125**]).

[105] A 4.1.4 (see note 32). Although Pāṇinīyas clearly distinguish grammatical gender from other notions of gender, they nevertheless consider it a property of what a nominal denotes; see PWT 183 84 [**273**]

[106] A 6.1.101; see note 32.

[107] A .3.114: *sarvanāmnaḥ syāḍ ḍhrasvaś ca* (PWT 325 [**499**]).

[108] A 6.3.35: *tasilādiṣv ā kṛtvasucaḥ* (PWT 256 [**369**]).

3 Lexicon

Pāṇini's full grammar (*śabdānuśāsana*) includes, in addition to the eight chapters of rules that constitute the *Aṣṭādhyāyī*, two ancillaries that can be considered types of lexicons: the dhātupāṭha and gaṇapāṭha. These also include sūtras (see PWT 85-135 [**132-211**]). Now, the gaṇa accompanying A 4.1.4 (see note 32) lists items like *ajā*, which represent the result of introducing the affix ***ṭāp*** and replacing contiguous *-a-ā-* with a single *-ā-* (see PWT 131 [**203**]). Similarly, the gaṇa accompanying A 3.3.104: *ṣidbhidādibhyo'ṅ* (see PWT 196 [**291**]) lists derivates *bhidā* and so on, which result from introducing ***aṅ*** to a verbal base, then introducing the feminine suffix ***ṭāp*** after the nominal base thus derived, and replacing *-a-ā* with *-ā-*: *bhid-a* → *bhid-a-ā* → *bhidā* (see PWT 203 [**298**]). The difference between such derivates and those like *tā-* (§2.2.4), which are clearly formed in the course of syntactic derivation, can be accounted for in general by distinguishing between qualifiers (*viśeṣaṇa*) and qualificands (*viśeṣya*). A qualifier takes a particular gender by virtue of being coreferential with a qualificand and thereby receives a feminine suffix in the course of derivation. A qualificand, on the other hand, has its gender from the outset. Accordingly, a term like *śuklā* as in (2) (§1.1) would now be derived from *śukla* in the course of syntactic derivation, but a nominal like *aṭavī* as in (52)-(54) (§2.2.4) would not.

One must keep in mind that Pāṇini's gaṇapāṭha does not include rules of derivation to form compounds or derivates with taddhita affixes. As shown (§2.2.3.3), optional compound formation is carried out regularly in the context of syntactic derivation of utterances: related padas in posited initial strings, from which are derived utterances with separate padas, are optionally combined.[109] The same holds for derivates like *tatra*, with taddhita affixes (§2.2.3.3). On the other hand, taddhita formations like *mathurāvat* (§2.2.2.2) differ. One can justifiably maintain that ***vati*** is introduced after *mathurā-i* of an initial string (24) *mathurā-i iva-s srughna-i prākāra-s.* This is in conformity with the fact that Pāṇini explicitly states that ***vati*** optionally follows a pada that is a value of *tatra*, that is, one that includes a seventh-triplet ending, and that such an ending is introduced in the context of deriving an utterance. On the other hand, the optional affixation does not also allow a string in which both *mathuvāvat* and *iva* occur: the suffix ***vati*** takes on the meaning of the particle, which is not used. Accordingly, the taddhita derivate may correctly be considered to reenter the derivation. Nevertheless, one cannot conclude that the formation of such a derivate takes place in any lexicon that is totally separate from the syntactic machinery.

4 Summary

In Pāṇini's system, padas of utterances are accounted for through a derivational procedure that starts from meaning conditions involving related actions and participants that serve to accomplish actions (*kāraka*) as well as things related to each other. It is undeniable that he thereby adopts the viewpoint that the derivational system serves to

---

[109] I have not taken up obligatory compounds; these call for special discussion, the details of which cannot be dealt with here.

form utterances (*vākyasaṁskāra*), not isolated words (*pada*) that are then strung together to form utterances (see §§1.2-1.4). Moreover, Pāṇini's derivational system may be aptly described as a continuum. Starting from meaning and cooccurrence conditions that determine the introduction of affixes to bases (§2.1), initial strings like (35) (§2.2.3.2) are formed. With the subsequent application of additional affixation and replacement rules, final strings like (20) (§2.2.2.1) are derived. The Pāṇinian derivational system does not absolutely separate morphology as a distinct component totally independent of syntax. On the contrary, what western grammarians call declensional and conjugational morphology are part of syntactic derivation (§2.2.1), and what is commonly called derivational morphology is also generally incorporated in the syntactic machinery. Thus, forms with what are called primary derivational affixes and which Pāṇini names *kṛt*, such as the participial suffixes ***ktavatu*** and ***kta*** in forms like *kṛtavān* and *kṛtaḥ* of (21) and (23) (§2.2.2.1), are introduced in the course of syntactic derivation. In addition, 'secondary derivation affixes', which Pāṇini calls *taddhita*, are regularly introduced after padas—terms that contain endings introduced in syntactic derivation—and not after mere bases in a separate morphological component (§2.2.2.2). Further, compounds, such as *rājapuruṣa-* 'king's servant' are formed from related padas such as *rājan-as puruṣa-s* of initial strings like (32) (§2.2.2.3), and it is proper to conclude that Pāṇini posited distinct initial strings such as (32) *rājan-as puruṣa-s prāsāda-i ni-s vas-l*, (40) *rājan-os puruṣa-s prāsāda-i ni-s vas-l*, and (41) *rājan-ām puruṣa-s prāsāda-i ni-s vas-l* to account not only for utterances like (30) *rājñaḥ puruṣaḥ prāsāde ni vasati*, (33) *rājñoḥ puruṣaḥ prāsāde ni vasati*, (34) *rājñām puruṣaḥ prāsāde ni vasati* but also for (31) *rājapuruṣaḥ prāsāde ni vasati*, in which the compound *rājapuruṣaḥ* occurs (§§2.2.2.3-2.2.3.3). Although composition supplies a new derived nominal base that enters into the derivation and receives a nominal ending, this operation does not take place in a separate morphological component but within the context of syntactic derivation. Moreover, even the formation of certain items with feminine suffixes, such as *tā* of *tasyām* in (53) *tasyām aṭavyāṁ vatsyāmi* (§2.2.4), takes place within the syntactic machinery and not in a totally separate lexicon.

In sum, Pāṇini presents us with a derivational system in which a sharp dichotomy between what western grammarians call syntax and morphology is absent. We have instead a beautifully integrated system accounting for utterances of Sanskrit.

Abbreviations

[For bibliographic details, see G. Cardona, *Pāṇini, A Survey of Research* The Hague-Paris: Mouton. Indian edition: Delhi: Motilal Banarsidass, 1980; reprinted with corrections and additions, 1997; G. Cardona, *Recent Research in Pāṇinian Studies*, second edition, Delhi: Motilal Banarsidass, 2004.]

A Pāṇini's Aṣṭādhyāyā

Bh. Patañjali's Mahābhāṣya (volume, page, line in Kielhorn's edition revised by K. V. Abhyankar/volume, page in the Rohtak edition)

HemAbh. Hemacandra's Abhidhānacintāmaṇi (kāṇḍa, verse in Abhidhānacintāmaṇi of Śrā Hemacandrāchārya , Edited ... by Śrā Haragovinda Śāstrā [Vidyabhawan Sanskrit Series 109], Varanasi: Chowkhamba, 1964)

Kāś Kāśikāvātti of Vāmana and Jayāditya (adhyāya, pāda, sūtra in Osmania University edition)
N Jinendrabuddhi's Kaśikāvivaraṇapañjikā, alias Nyāsa, on the Kāśikāvṛtti (adhyāya, pāda, sūtra, volume, page in edition by Dwarika Das Shastri and Kalika Prasad Shukla)
PM Haradatta's Padamañjarī on the Kāśikāvṛtti (adhyāya, pāda, sūtra, volume, page in edition by Dwarika Das Shastri and Kalika Prasad Shukla)
PK Rāmacandra's Prakriyākaumudī (volume, page of K. P. Trivedi's edition)
Pr. Kaiyaṭa's Mahābhāṣyaradāpa (volume, page of Rohtak edition/volume, page, column of Nirṇaya Sāgara Press edition)
PWT Cardona, G.: Pāṇini, his Work and its Traditions, 2nd edn. Background and Introduction, vol. I. Motilal Banarsidass, Delhi (1997)
RA Dharmakīrti's Rūpāvatāra (volume, page of Rao Bahadur M. Rangacharya's edition)
SK Bhaṭṭoji's Siddhāntakaumudī (volume, page of edition by Giridhara Śarmā Caturveda and Parameśvarānanda Śarmā Bhāskara)
Ud. Nāgeśa's Mahābhāṣyapradāpodyota (volume, page of Rohtak edition/volume, page, column of Nirṇaya Sāgara Press edition)
VBh. Kauṇḍabhaṭṭa's Vaiyākaraṇabhūṣaṇa (kārikā, page in Manudeva Bhaṭṭāchārya's edition)
VBhS Kauṇḍabhaṭṭa's Vaiyākaraṇabhūṣaṇasāra (kārikā, page in Bala Krishna Pancholi's edition)
VP Bhartṛhari's Vākyapadāya (kāṇḍa, kārikā [kāṇḍa, samuddeśa kārikā for third kāṇḍa] of W. Rau's edition, 1977)
VPA Raghunātha Sharma's Ambākartrī commentary on the Vākyapadīya and its commentaries
VPH Helārāja's Prakīrṇaprakāśa on the third kāṇḍa of the Vākyapadāya (kāṇḍa, samuddeśa, kārikā, page, lines in K. A. Subramania Iyer's edition, preceded by kāṇḍa, samuddeśa, kārikā numbers from Rau's edition)
VPP Śrīvṛṣabha's paddhati, the Sphuṭākṣara, on the Vākyapadīya and its Vṛtti (page, lines in K. A. Subramania Iyer's edition)
VPVṛ. Bhartṛhari's autocommentary (vṛtti) on the Vākyapadāya (kāṇḍa, kārikā, page, lines in K. A. Subramania Iyer's editions)
vt. vārttika

On the Architecture of Pāṇini's Grammar*

Paul Kiparsky

Stanford University

Abstract. The descriptive technique of Pāṇini's *Aṣṭādhyāyī* has deeply influenced modern linguistic theory and practice. But Pāṇini had no predetermined "theory of grammar". The rich array of formal devices and categories of his analysis emerge from nothing more than rigorously making it as short as possible. I review the overall organization of his grammar, the types of rules and the categories that they apply to, and the principles which govern their application and interaction in the system. I present in outline Pāṇini's treatment of the major topics of Sanskrit grammar: syntax, word-formation (primary and secondary derivation, compounding), nominal and verbal inflection, and phonology. The analysis reveals that central aspects of what linguistics ascribes to Universal Grammmar, including levels of representation, thematic roles, and rule ordering, are motivated within the *Aṣṭādhyāyī* merely by the goal of achieving maximum compression. This finding is itself of considerable theoretical interest.

Keywords: Pāṇini, Sanskrit grammar, syntax, morphology, phonology, simplicity, thematic roles, levels of representation, rule ordering.

1 Organization of the Grammar

1.1 Introduction

Pāṇini's grammar is universally admired for its insightful analysis of Sanskrit. In addition, some of its features have a more specialized appeal. Sanskritists prize the completeness of its descriptive coverage of the spoken standard language (*bhāṣā*) of Pāṇini's time, and the often unique information it provides on Vedic, regional and even sociolinguistic usage.[1] Theoretical linguists of all persuasions are in addition impressed by its remarkable conciseness, and by the

* This material was presented at the CIEFL conference on the Architecture of Grammar organized by K.G. Vijayakrishnan and his colleagues in Hyderabad 15-17.1.2002, and published as a little booklet on that occasion. Portions were also presented in a series of lectures at UCLA in March 2002. My thanks to the audience at these talks, as well as to Jan Houben and Gérard Huet for their thoughtful comments. I am solely responsible for any remaining errors in this revised text.

[1] Whitney's attempt to discredit Pāṇini on matters of fact was almost unanimously repudiated by Sanskritists (Bühler 1894, von Schroeder 1895, Thieme 1935). The better we come to understand Pāṇini's rules, the more their accuracy is vindicated (see e.g. Kiparsky 1979:13.)

G. Huet, A. Kulkarni, and P. Scharf (Eds.): Sanskrit CL 2007/2008, LNAI 5402, pp. 33–94, 2009.

rigorous consistency with which it deploys its semi-formalized metalanguage, a grammatically and lexically regimented form of Sanskrit. Empiricists like Bloomfield also admired it for another, more specific reason, namely that it is based on nothing but very general principles such as simplicity, without prior commitments to any scheme of "universal grammar", or so it seems, and proceeds from a strictly synchronic perspective. Generative linguists for their part have marveled especially at its ingenious technical devices, and at intricate system of conventions governing rule application and rule interaction that it presupposes, which seem to uncannily anticipate ideas of modern linguistic theory (if only because many of them were originally borrowed from Pāṇini in the first place).

This universal admiration of Pāṇini poses a problem. Why do linguists who don't approve of each other nevertheless agree in extolling Pāṇini? Each school of linguistics seems to fashion its own portrait of Pāṇini. In the following pages I propose to reconcile the Bloomfieldian portrait of Pāṇini with the generative one by showing how the grammar's extremely rich formal principles and thematic groupings of rules emerge from nothing more than rigorously requiring the description to be as simple as possible. To this end, I discuss some of the aspects of the *Aṣṭādhyāyī* that are of particular linguistic interest. These include insights into the organization of grammar, as well as descriptive generalizations that either support or call into question certain contemporary theories. I will begin with an exposition of some of the general features of the grammar and then examine in turn its "syntax", "morphology", and "phonology", on the understanding that these are themselves emergent constellations of rules rather than predetermined components into which the description is organized.

Needless to say, this perspective compels me to be highly selective, and to set aside a vast number of technical aspects that are crucial to a deeper understanding of the system. Although the text of the grammar is probably preserved rather well, it requires interpretation. Some of the principles that determine the application of its rules are not stated in the grammar itself, and must be inferred. A massive commentatorial tradition is concerned with just this, offering ingenious (and it must be said, sometimes too ingenious) criticism and justification of the wording of Pāṇini's rules. Modern research is complementing this with a reconstruction of Pāṇini's grammatical thinking in historical perspective, with a view to determining the structure of the grammar through internal analysis, using actual Sanskrit usage to help settle interpretive dilemmas about the precise intent of rules, and gleaning additional hints from phonetics, ritual, and other related areas. In this field it is unfortunately impossible to avoid controversy, and the reader should realize that practically everything that follows is subject to challenge.

A final note of caution before we move on. The fact that Pāṇini studied a single language, with simplicity as the guiding principle of the analysis, has certain corollaries which must be appreciated if we are not to pass anachronistic judgments on the grammar. Some formulations in the *Aṣṭādhyāyī* reflect what we might regard as purely notational simplifications which do not correspond to any linguistically significant generalizations. For example, the word order of rules is usually chosen so as to maximize syllabic contraction between words. Almost all the genuine

generalizations are there, but so are some spurious one. This is the inevitable result of adopting a formal economy principle in the absence of cross-linguistic criteria for determining the substantive content of notational conventions.

1.2 Components

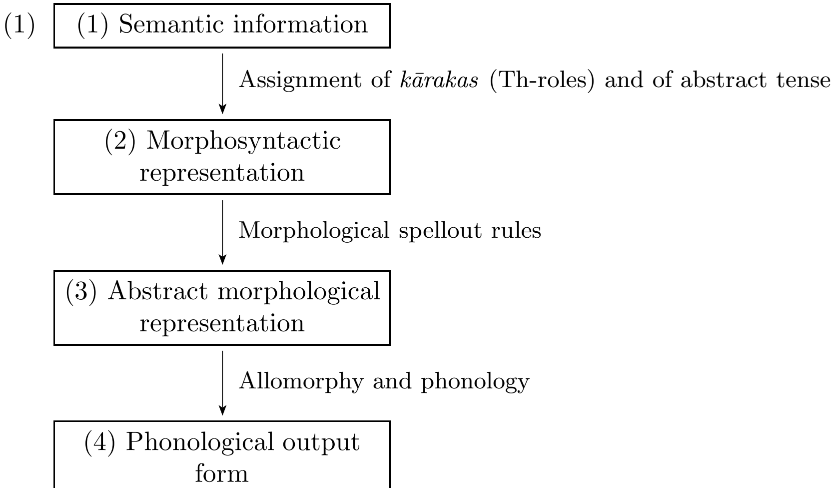
The grammar has four distinct components:

a. *Aṣṭādhyāyī*: a system of about 4000 grammatical rules.
b. *Śivasūtras*: the inventory of phonological segments, partitioned by markers *(anubandhas)* to allow classes of segments *(pratyāhāras)* to be designated by a set of special conventions.
c. *Dhātupāṭha*: a list of about 2000 verbal roots, with subclassification and diacritic markers encoding their morphological and syntactic properties.
d. *Gaṇapāṭha*: a list of 261 lists of lexical items which are idiosyncratically subject to certain rules. Some of the lists are open-ended, and (like the *Dhātupāṭha*) they have to some extent been modified by later grammarians.

The rules of the *Aṣṭādhyāyī* make reference to classes defined on the elements in the other three components by means of conventions spelled out by rules of the *Aṣṭādhyāyī* itself. Thus, while none of the components is intelligible in isolation, together they constitute a complete integrated grammar and lexicon.

1.3 Levels

The rules of the *Aṣṭādhyāyī* fall into three broad classes, each of which effects a mapping between what we (from a somewhat anachronistic modern perspective) could see as different levels of representation. Thus, the grammar analyzes sentences at a hierarchy of four levels of description, which are traversed by three mappings in the direction from semantics to phonology (Kiparsky & Staal 1969, Bronkhorst 1979, Joshi & Roodbergen 1980).

(1)


This is the top level of articulation of the grammar. Each part has a further rich internal organization, as described below.

Extending and modifying a discussion by Houben 1999, I should like to make more precise what the interface between the levels looks like, and why it is directional. Consider the sentence whose output (phonological) form is shown in (2):

(2) *vánād grā́mam adyópétyaudaná āśvapaténā́pāci*
'When Āśvapata came from the forest to the village today, he cooked some rice.'
(*literally:*) 'Having come..., rice was cooked by Ā."

Retracing its derivation backwards, the morphological analysis is as follows (capitals are silent diacritics that show grammatical properties):

(3) **vána-N̄asI grā́ma-am adyá upa-ā-iṆ-Ktvā odaná-sU**
forest-AblSg village-AccSg today Prep-Prep-go-Abs rice-NomSg
áśva-páti-aṆ-Ṭā á-ḌUpacAṢ-CiṆ-ta
A.-descendant-InstrSg Aug-cook-AorPass-3Sg

At the next higher, morphosyntactic level, the arguments are assigned thematic roles *(kārakas)* and the root is assigned abstract tense, in this case *luṅ* 'aorist':

(4) a. Sentence 1:
   (1) *vána* 'forest': source *(apādāna)*
   (2) *grā́ma* 'village': goal *(karman)*
   (3) *adyá* 'today': temporal location *(adhikaraṇa, kāle)*
   (4) *āśvapatá* 'descendant of *Aśvapati*': agent *(kartṛ)*
   (5) *upa-ā-iṆ+Ktvā* 'approach, reach', absolutive
 b. Sentence 2:
   (1) *odaná* 'rice': goal *(karman)*
   (2) *āśvapatá* 'descendant of *Aśvapati*': agent *(kartṛ)*
   (3) *ḌUpacAṢ+luṅ* 'cook', aorist tense

The derivation is initiated by constructing the morphosyntactic analysis (4) on the basis of the ontology of the events and the speaker's wish to express certain features of it *(vivakṣā)*. This initial stage includes the following steps:

(5) a. The participant which is independent bears the role of agent *(kartṛ)*.
 b. When there is a separation, the participant which is fixed bears the source role *(apādāna)*.
 c. The participant which is primary target of the action bears the goal/ patient role *(karman)*.
 d. Roots denoting recent past events (events which have occurred previously on the present day) are assigned Aorist tense *(luṅ)*.
 e. The suffix *Ktvā* is assigned (instead of aorist tense) to express to the root denoting the prior event if it has the same agent as the posterior event.

Formally, (5a-c) are definitions.

The grammar is a device that starts from meaning information such as (5) and incrementally builds up a complete interpreted sentence. A representation at a given level can be mapped, usually in more than one way, into a representation at the next level, which incorporates all the information accumulated in the derivation so far. For example, a given propositional content can be expressed in several different ways at the next level as a semantically interpreted morphosyntactic representation. The final result is a phonetic representation with an associated full semantic, morphosyntactic, and morphological interpretation.

The mapping between each successive pair of levels is constrained by the representations at those levels and at earlier levels. Later levels play no role. So, the morphology-to-phonology mapping (level 3 → level 4) in (1)) responds to semantic and syntactic factors (levels 1 and 2), but the semantics-to-morphosyntax mapping (1 → 2 in (1)) never cares about phonology or morphology factors (levels 3 and 4). And the morphosyntax-to-morphology mapping (2 → 3) is sensitive to semantics (level 1), but not to phonology (level 4).

Moreover, the morphology/phonology mapping (3 → 4) differs fundamentally from the others. It is the only one which allows destructive (non-monotonic) operations such as substitution and deletion. Both morphological elements and phonological segments may be replaced by other morphological elements or segments, or by zero. (The decision to treat allomorphy as replacement was a fateful one, as we shall see.) These processes create extensive opacity, i.e. application of rules in non-surface-true contexts, which forces use of a Pāṇinian counterpart to extrinsic ordering. In contrast, the mappings between the other levels are strictly monotonic. For example, there are no processes which delete *kārakas* or abstract tenses, or which replace one *kāraka* or abstract tense by another. In this system, it is really true that "phonology is different" (Bromberger and Halle 1989).

The two asymmetries just outlined imply an intrinsic directionality of Pāṇinian derivations. They cannot run in the other direction, because the required information about higher-level representations is not available at lower levels. Both the top-to-bottom nature of conditions on rule application, and the non-monotonic character of the morphology/phonology mapping, make derivations irrecoverable.

An example of the "top-down" kind of rule which does *not* occur would be: "The agent of a gerundive is assigned genitive case if it bears initial accent." What *does* occur are phonological rules (3 → 4) which are sensitive to semantic or thematic information (levels 1, 2), such as:

(6) 6.2.48 तृतीया कर्मणि
tṛtīyā karmaṇi (45 kte ca) (6.2.1 prakṛtyā pūrvapadam)
third-Nom *karman*-Loc
'The first member of a compound with a [deleted] instrumental case keeps its original accent if the second member is a past participle that denotes the Goal (i.e. passive)'

According to this rule, we have *áhi-hata* 'killed by a snake' (passive, initial accent), but *ratha-yātá* 'gone by cart' (active, with default final accent). The morphological representation of the compounds is parallel. Both have a first member in the instrumental case, and a second member in the participle suffix -*Kta*.

(7) a. **ahi-Ṭā han-Kta-sU**
snake-Instr kill-PP-Nom

b. **ratha-Ṭā yā-Kta-sU**
cart-Instr go-PP-Nom

They are differentiated only at the morphosyntactic level. In (7a), instrumental case expresses the agent role and the participle suffix *-Kta* expresses the goal/patient role. In (7b), instrumental case expresses the instrument role and the participle suffix *-Kta* expresses the agent role. Instrumental case expresses either of these roles by rule (8):

(8) 2.3.18 कर्तृकरणयोस्तृतीया

kartṛkaraṇayos tṛtīyā
agent-instrument-LocDu third-Nom

'The third (triplet of case endings) expresses *kartṛ* or *karaṇa*'.

After *yā* 'go', the suffix *-Kta* can be either active or passive (i.e., personal as well as impersonal):

(9) 3.4.72 गत्यर्थाकर्मकश्लिषशीङ्स्थासवसजनरुहजीर्यतिभ्यः

gatyarthākarmakaśliṣaśīṅsthāsavasajanaruhajīryatibhyaḥ
go-meaning-intrans-*śliṣa*... AblPl

'*Kta* expresses agent, process, and goal/patient, after verbs meaning "go", intransitives, *śliṣa* 'embrace', ... '

(How this works exactly will be explained in section 1.4.1.)

The upshot is that accent in this class of compounds depends not on the morphology (not even on the abstract morphological representation prior to the deletion of stem-internal case endings), but on the thematic relations that the morphology expresses. Such examples constitute a potential challenge to theories of grammar in which phonology is a module that does not have access to thematic role information. A project for the future is to determine whether such data can be reconciled with a modular view of grammar. In this case, a solution might be to treat the accentuation of compounds as part of their morphological composition, and to view the input to compounding as a thematic/semantic representation (rather than as a syntactic structure, as Pāṇini does).

The organization into "levels" described here flows from the strict adherence to economy. It would be misleading to say that (1) reflects a "theory of grammar". It is just the simplest and most efficient way to describe the language. We might want to take this as evidence that language is "really" organized that way, but we would be taking that step on our own as theoretical linguists.

Like the levels themselves, the directionality was not necesarily assumed in constructing the system either. More likely, it is another emergent property of the analysis, which reflects a real asymmetry of language. Let us see how this might be the case.

1.4 How Simplicity Dictates the Form and Grouping of Rules

1.4.1 Headings and Gapping

Almost all technical aspects of the grammatical system are motivated ultimately by the fundamental requirement of simplicity (economy, *lāghava*). Simplicity dictates the formation of technical terms, many basic analytic decisions, and the wording and grouping of rules. Consider the formation of patronymics — derived nouns that designate a descendant of the person named by the base. A descendant of Upagu is called *Aupagavá*, formed by the suffix *-aṆ*, phonologically *-a*, with a diacritic *Ṇ* which causes *vṛddhi* strengthening of the stem's initial syllable; general rules accent the suffix, and truncate the stem-final *-a* before it. With the other patronymic suffixes, *-aṆ* constitutes a subclass of *taddhita* suffixes ('secondary' denominal derivational suffixes), which share a number of properties: they are added to nominal stems *(prātipadikas)*, and they are optional, in the sense that there is a synonymous analytic expression: *Aupagavá* is synonymous with *Upagor apatyam* 'Upagu's descendant'. *Taddhita* suffixes in turn are a subclass of suffixes *(pratyaya)*, which share some more general properties, such as being placed after the base, and being accented (in the default case). Finally, suffixation is a subtype of word-formation, and all such processes share the property of applying only to semantically related elements: the two conjoined phrases *kambalam upagor, apatyaṃ devadattasya* 'the blanket of Upagu, the descendant of Devadatta' do not correspond to **kambalam Aupagavo devadattasya* 'the blanket Aupagava Devadatta'.

Such shared properties are not repeated for each suffixation rule; they are stated just once in a heading with the appropriate scope. Suffixes — essentially the items introduced in books 3–5 of the grammar — are governed by the headings (10) and (11).

(10) 3.1.1 प्रत्ययः
pratyayaḥ
suffix-Nom
'(an item introduced in the rules up to the end of 5) is (termed) *pratyaya* "suffix"'

(11) 3.1.2 परश्च
paraś ca
following-Nom and
'and (an item introduced in the rules up to the end of 5) follows'

(12) 3.1.3 आद्युदात्तश्च
ādyudāttaś ca
initial-accent-Nom and
'and has initial accent'

Rule [12] causes suffixed forms to be accented on the first syllable of their (last added) suffix. This is merely the default case which is realized if none of the more specific accent rules is applicable. There are many classes of suffixes with special accentual properties and many special rules for the accentuation of compounds and other derived words, which have priority over [12].

The other, more specific properties of *taddhita* suffixes and of their patronymic subclass are specified by rules that head the rules that introduce these elements.

(13) 4.1.1 ङ्याप्प्रातिपदिकात्
ṅyāpprātipadikāt
Ṅī-āP-stem-Abl
'after (an item ending in the feminine suffixes) *Ṅī*, *āP*, or (after) a nominal stem'

(14) 4.1.76 तद्धिताः
taddhitāḥ
taddhita-NomPl
'denominal suffixes'

(15) 4.1.82 समर्थानां प्रथमाद्वा
samarthānāṃ **prathamād vā**
semantically-related-GenPl first-Abl optionally
'After the first semantically related stem [marked by a pronoun in the genitive case in each rule], optionally [preferably].'

All these headings contribute to the rule that suffixes -*aṆ*:

(16) 4.1.83 प्राग्दीव्यतो ऽण्
prāg dīvyato 'ṇ
up-to *dīvyati*-Abl *aṆ*-Nom
'Up to rule 4.4.2, the accented *taddhita* suffix *aṆ* is added after the first semantically related nominal stem [marked by a pronoun in the genitive case in each rule].'

The meaning of *aṆ* is specified separately in a later rule. Since most *taddhita* suffixes are polysemous, and complex sets of meanings may be shared by several suffixes, the assignment of meanings to *taddhitas* is decoupled from their affixation. The meaning rules are interspersed with the suffixation rules in such a way as to permit the maximally simple wording of each. For example, the basic meaning of the several patronymic suffixes is assigned jointly by rule (17):

(17) 4.1.92 तस्यापत्यम्
tasyāpatyam
his descendant
'(The suffixes *aṆ* etc.) denote 'descendant' [of the stem corresponding to the genitive pronoun in the rule].'

It should be clear from these examples that the rules in text as it stands are not intelligible on their own. Rather, a rule is a condensed formulation from which a complete rule may be constructed by following certain interpretive procedures.

Throughout the grammar, rules of the same type are grouped together and the expressions which they have in common are omitted from each individual rule and stated once and for all as a heading at the beginning of the group. Every rule that comes under the heading must then be understood as implicitly including the expression in the heading, in so far as it is compatible with the rule's wording. Such major headings divide the *Aṣṭādhyāyī* into overlapping topical sections. In principle any shared property (trigger, target, intervening

element, other conditions) may be the basis for such a grouping of rules. For the phonological rules, for example, the most important headings have to do with two kinds of shared properties: the DOMAINS in which rules are applicable, and the INTERACTION of rules in derivations.

Rules get their full meaning not just from special headings, but from other ordinary derivational rules, by a technical generalization of the "gapping" processes of natural language (*anuvṛtti*, Joshi and Bhate 1984).

(18) GAPPING: An element of a rule is continued in the next rule if it is compatible with it. When an element is discontinued, all its modifiers are also discontinued.

Since gapping requires adjacency, related rules must be grouped together so that their shared parts can be factored out.

1.4.2 Blocking

Simplicity leads directly to another device which determines the organization of the whole grammar, BLOCKING *(utsarga/apavāda)*. By convention, special rules block incompatible general rules, which allows a simpler (and more natural) wording of the general rule.

(19) BLOCKING: If the domain of rule A is properly included in the domain of rule B, and both rules cannot apply, then A blocks rule B in the overlapping domain.

The rules so related need *not* be grouped together in the grammar. The grouping of rules is instead designed to maximize gapping and the generality of headings. In this way, there emerges a clear topical organization of the grammar. E.g. suffixation processes occupy a continuous block of rules (books 3-5 in the conventional numbering scheme, which is probably not original).

To see the interplay between gapping and blocking in topical groups of rules on a small scale we turn again to *-aṆ*. It is just the most general ("elsewhere") patronymic suffix. Various classes of stems require other patronymic suffixes in their place, which have the same meanings and all the other general properties that we listed. For example, a class of stems, among them those ending in *-pati* 'lord', form their patronymics with the suffix *-Ṇya*, e.g. *Prajāpati* → *Prājāpatyá*. This exception has in turn an exception: a class of compounds in *-pati*, listed in the *Gaṇapāṭha*, require not *-Ṇya*, but *-aṆ* again, e.g. *Aśvapati* → *Āśvapatá*. Since these rules all come under the sway of (10)–(17), they must be grouped together in their scope. Maximal concision is achieved by grouping together the two *-aṆ* rules, the general case and the exception to the exception, followed by the *-Ṇya* rule, the first-order exception. That way *-aṆ* has to be repeated just once. So in the grammar rule (16) is immediately followed by:

(20) 4.1.84 अश्वपत्यादिभ्यश्च
aśvapatyādibhyaś ca (83 aṆ) (82 samarthānāṃ . . .) . . .
'*Aśvapati*-etc.-Abl
'The *taddhita* suffix *-aṆ* is also added after the first syntactically related stem which belongs to the class *Aśvapati* etc.' [Exception to 4.1.85].

(21) 4.1.85 दित्यदितिआदित्यपत्युत्तरपदाण्ण्यः
dityaditiādityapatyuttarapadāṇ ṇyaḥ **(82 samarthānāṃ . . .)**
diti-aditi-āditya-pati-second-word-Abl ṇya-Nom
. . .

'The *taddhita* suffix *-Ṇya* is added after the first syntactically related stems *Diti* . . . and after compounds in *-pati*.'

The example seems absurdly simple, but it represents a common situation that already exceeds the expressive power of some modern theories designed to deal with networks of related morphological processes and "elsewhere" relations. The device of inheritance hierarchies, for example, establishes a nesting of subclasses, where each class specifies its own special properties and otherwise inherits the properties of its mother class by default. In such a hierarchy, the suffix *-aṆ* and its properties must be listed twice (Deo 2007).

(22)
-aṆ
|
-Ṇya
|
-aṆ

The *-aṆ* that is an exception-to-an-exception cannot inherit its properties from the regular (default) *-aṆ* at the top of the hierarchy across the intervening *-Ṇya*, so the fact that the two *-aṆs* are identical in all their morphological idiosyncrasies is an accident.

Morphologists have identified two types of blocking, FORMAL and SEMANTIC. In formal blocking, a morphological process, typically paradigmatic, is blocked by the existence of a more specific process that conveys the same meaning, as in our patronymic example. (An English illustration is the blocking of word-level plural **mans* by the stem-level plural *men*.) In semantic blocking, a word with a specialized meaning delimits the meaning of a (formally related or unrelated) derived word. The instrumental meaning of *-er*, which is blocked by the existence of a more specialized word denoting the instrument. E.g. a *cutter* is any cutting implement other than a knife, scissors, adze, chisel, axe, or other specially designated item, and a *sweeper* is any sweeping implement other than a broom, brush, or other specially designated item. In modern Greek, the regular diminutive/hypocoristic meaning of the productive suffix *-aki* is blocked by idiomatic meanings of words in *-aki*: because *sakaki* means 'coat', its compositional meaning 'little sack' is blocked.

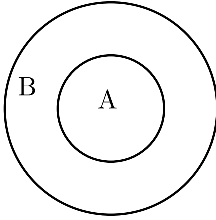
For Pāṇini, formal and semantic blocking are not symmetrical. Since affixes expressing the same meaning compete with each other, formal blocking is an automatic consequence of the principles of the grammar. Therefore, where blocking does not actually obtain — that is, when there in fact are two derivatives with the same meaning — the rule introducing one of them must be explicitly designated as optional. In the case of semantic blocking, the default is just the opposite, for meanings do not formally compete with each other in the system. Therefore,

when semantic blocking actually obtains, a special rule or condition is required. For example, the intensive suffix *yaṄ* is added to roots to express repetition *(kriyāsamabhihāra*, 3.1.23). With certain roots, *yaṄ* is added in an idiomatic meaning: with roots meaning 'go' it denotes a crooked manner, e.g. *cañkramyate* 'meanders' (3.1.24), and with roots such as *lup* 'cut', *sad* 'sit' it denotes a sloppy manner, e.g. *sāsadyate* 'slouches'. The point is that these special meanings block the general meaning of the intensive: *cañkramyate* and *sāsadyate* do not mean 'goes repeatedly', 'sits repeatedly'. Such semantic blocking is readily handled in Pāṇini's grammar, but requires an extra effort (in this case, the mention of *nityam* in the rule). A more central and systematic case of semantic blocking is the disjunctive assignment of thematic roles discussed below.

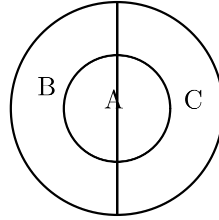
Collective blocking generalizes the blocking relation between special and general rules from pairs of rules to sets of arbitrary size. Blocking is then induced not only by a domain inclusion relationship between two rules, but also by a domain inclusion relationship between a rule and a *set* of rules.

(23) Collective blocking: If the domain of rule A is properly included in the union of the domains of a set of rules $\mathcal{R} = \{R_1, R_2, \ldots R_n\}$, A blocks each of the rules in $\mathcal{R}$.

So rule A blocks both B and C below:



Regular blocking



Collective blocking

An example of collective blocking are the rules that resolve sequences of adjacent vowels in close contact. (24) shows three processes, glide formation (rule (25)), monophthongization (rule (26)), and contraction (rule (27)), stated directly below:

(24) *adya iha apāci odanaḥ* → *adyehāpācyodanaḥ* 'the rice was cooked here today'

(25) 6.1.77 इको यणचि
iko yaṇ aci (72 saṃhitāyām)
ik-Gen *yaṇ*-Gen *ac*-Loc
'*i, u, ṛ, ḷ* → *y, v, r, l* before a vowel, in close contact.'

(26) 6.1.87 आद्गुणः
ād guṇaḥ (77 aci) (72 saṃhitāyām)
a-Abl *guṇa*-Nom
'*a* (short or long) and a vowel are (together) replaced by *guṇa (a, e, o)*, in close contact.'

(27) 6.1.101 अकः सवर्णे दीर्घः
akaḥ savarṇe dīrghaḥ (84 ekaḥ pūrvaparayoḥ) (72
aK-Gen same-color-Loc long-Nom
saṃhitāyām)

'*aK (a, i, u, ṛ, ḷ)* and a following vowel of the same color are (together) replaced by a long vowel, in close contact.'

There is no proper inclusion relationship between the environments of any two of the three rules [25], [26], and [27]. Rules [25] and [27] overlap for *i+i, u+u*, [26] and [27] overlap for *a+a*, and [25] and [26] have no common domain. If the rules are taken pairwise, there is no possibility of blocking here. But if we take the three rules together, it is evident that the input to [27] is a proper subset of the *combined* inputs to [25] and [26]. From this point of view, [27] has no domain of its own, and by generalized blocking should set aside each of the other two rules in their respective shared domains. This is the correct result, for it is true that [27] always wins both over [26] (*a+a* → *ā*, not *a*) and over [25] (*i+i* → *ī*, not *yi*).

These three rules are merely the top nodes of an intricate blocking hierarchy. The principal restriction on (26) is (28).

(28) 6.1.88 वृद्धिरेचि
vṛddhir eci (87 ād) (72 saṃhitāyām)
a-Abl *guṇa*-Nom
'*a* (short or long) and *e, o, ai, au* are (together) replaced by *vṛddhi (ā, ai, au)*, in close contact.'

E.g. *upetya odanaḥ* → *upetyaudanaḥ* (see (2)). (28) is in turn blocked by (29):

(29) 6.1.94 एङि पररूपम्
eñi pararūpam (91 upasargāt, dhātau) (87 āt) (72
eñ-Loc following-form-Nom
saṃhitāyām)

'*a* (short or long) followed by *e* or *o* in a root are (together) replaced by the second vowel.'

For example, *upa-elayati* → *upelayati (*upailayati)*. But (29) is itself blocked by (30):

(30) 6.1.89 एत्येधत्यूठ्सु
etyedhatyūṭhsu (88 vṛddhiḥ) (87 āt) (72 saṃhitāyām)
eti-edhati-ūṬh-LocPl
'*a* (short or long) and *e, o, ai, au* in (the roots of the verbs) *eti-edhati* and in (the replacement) *ūṬh* are (together) replaced by *vṛddhi (ā, ai, au)*, in close contact.'

E.g. *upa-eti* → *upaiti (*upeti)*. Finally, (31) blocks (30):

(31) 6.1.95 ओमाङोश्च
omāṅoś ca (94 eṅi pararūpam) (87 āt) (72 saṃhitāyām)
om-āṄ-LocDu and
'*a* (short or long) with (the first *e* or *o* of) *om* or *āṄ* are (together) replaced by the second vowel.'

E.g. *upa-ā-itya* → *upetya (*upaitya)* (see (2)).

By a similar reasoning, rule (32) collectively blocks rules (27) and (26), the latter vacuously.

(32) 6.1.97 अतो गुणे
ato guṇe (96 apadāntāt) (72 saṃhitāyām)
aṬ-Gen *guṇa*-Loc
'Short non-final *a* followed by *guṇa (a, e* or *o)* are (together) replaced by the second vowel.'

This gives e.g. *grāma-am* → *grāmam* in (2).

1.5 Word Integrity

The integrity of words appears not only in word-formation processes (p. 39, and see also (137) below), but also in phonology. The general principle is that phonological processes across word boundaries are "invisible" *(asiddha)* to word-internal phonological processes. This has two aspects. Phonological processes applicable within words have priority over phonological processes applying to combinations of words. And when the application of a process to a combination of words creates the conditions for the application of a process within a word, the word-internal process does not apply. (Translated into rule ordering terminology: rule applications across word boundaries do not feed or bleed rule applications inside words.) Let us illustrate these points with the phonological rules just discussed. The following pairs of words each show the sequence *a i i*, but it is resolved in two different ways depending on where the word boundary comes.

(33) a. *a+yaj+a+i indra-am* → *ayaja indram*
b. *atra i+ij+atuḥ* → *atrejatuḥ*

The right results are obtained if word-internal applications of the rules are given precedence over applications across word boundaries.

(34) Correct derivation of *á-yaj-a indram* 'I sacrificed to Indra'
á-yaj-a-i indra-am
á-yaj-e indra-am (26) 6.1.87 **ād guṇaḥ**
á-yaj-a indram (32) other rules

(35) Wrong derivation:
á-yaj-a-i indra-am
á-yaj-a īndram (27) 6.1.101 **akaḥ savarṇe dīrghaḥ**,
(32) 6.1.97 **ato guṇe**
*á-yaj-endram (26) 6.1.87 **ād guṇaḥ**

The following example shows how phrase phonology does not feed word phonology. By 7.4.59 **hrasvaḥ**, the vowel of a reduplicant is shortened. By the general convention 1.1.48 **eca ig ghrasvādeśe**, when *e* is shortened it becomes *i*. (36) shows that the *e* which arises by sandhi (which is part of both words by the convention 6.1.85 **antādivacca**) does not shorten:

(36) Right and wrong derivations for 'he sacrificed here'
atra i-yā́j-a
atreyā́ja (correct output) (26) 6.1.87 **ād guṇaḥ**
*atriyā́ja 7.4.59 **hrasvaḥ**

Similarly, from *pacāva idam* 'let us two cook this!' the vowel contraction rule *ād guṇaḥ* derives *pacāvedam*, where the *e* does not undergo rule 3.4.93 **eta ai**, which turns *e* into *ai* in imperative suffixes (as in 1.Du. *pacāvahai*).

(37) Right and wrong derivations for 'let us two cook this!'
pácāva idam
pácāvedam (correct output) (26) 6.1.87 **ād guṇaḥ**
*pácāvaidam 3.4.93 **eta ai**

1.6 The *Siddha*-Principle

The fundamental principle governing the interaction of rules in the grammar is the *siddha*-principle:

(38) सर्वत्र सिद्धवत्
sarvatra siddhavat
everywhere effected-like
'(Any rule) is treated as having taken effect when applying any (rule).'

The *siddha*-principle has a positive and a negative aspect. The positive aspect is that, if A creates inputs to B, the effects of A are taken into consideration when applying B (in traditional terminology, *ādeśalakṣaṇabhāva* 'giving scope to an operation conditioned by the input'; in ordering terminology, A takes effect before B, FEEDING ORDER). The negative aspect is that, if A deprives

B of inputs, the effects of A are also taken into consideration when applying B (*utsargalakṣaṇapratiṣedha* 'prohibition of an operation conditioned by the input', or equivalently, A precedes B, BLEEDING ORDER). What the *siddha*-principle says is that rules interact in a TRANSPARENT way, under the slogan "environment-changing rules first".

The *asiddha* relation can be defined as follows.

(39) In a derivation where rules A and B are applied to ϕ in that order, rule A is *asiddha* if and only if the result is identical to the result of applying A and B to ϕ simultaneously, and distinct from the result of applying them to ϕ in the opposite order.

The *siddha*-principle is a default principle which can be defeated at cost. The usual method of defeating it is to declare a rule *asiddha* '(treated as) not having taken effect, suspended, invisible' with respect to the rule that it makes opaque. In a smaller group of cases, where bleeding must be blocked but feeding is still allowed, a different method is used: the output of the opacity-causing rule is declared to be *sthānivat* '(treated) like the input'.

Most rules which must be prevented from feeding or bleeding other rules are collected into two special sections whose headings stipulate *asiddhatva* (the *asiddha* property) for all of them. The most important group extends from 8.2.1 though the last three sections of the grammar, and is called the *Tripādī* 'Three Sections'. It is headed by one of the most famous rules of the Aṣṭādhyāyī:

(40) 8.2.1 पूर्वत्रासिद्धम्
pūrvatrāsiddham
before non-effected
'(any rule in this section is treated as) not having taken effect when applying any previous (rule of the grammar)'

The effect of (40) is to fix the application of the rules in the last three books to the order in which they are enumerated in the grammar (modulo blocking of general rules by special rules). The rules in this section, therefore, are organized like the rules of a classical generative phonology.

For example, in the derivation of Instr.Pl. *rājan-bhis* → *rājabhiḥ* 'by the kings', the final *-n* of the underlying stem *rājan-* is deleted by rule (41).

(41) 8.2.7 नलोपः प्रातिपदिकान्तस्य
nalopaḥ prātipadikāntasya (8.1.16 padasya)
n-null-Nom stem-end-Gen
'stem-final *n* is deleted (at the end of a word)'

Rule 7.1.9 substitutes the suffix *-ais* for *-bhis* after short *a*, e.g. *vṛkṣa-bhis* → *vṛkṣa-ais (→ vṛkṣaiḥ)* 'by the trees'. Because (41) is in the *Tripādī* section and rule 7.1.9 is outside the *Tripādī* section, (40) correctly blocks (41) from feeding 7.1.9 in *rājabhiḥ*.

In the derivation of *pakva* ‘cooked’ from underlying *pac-Kta*, two processes are applicable. Rule 8.2.52 requires the replacement of *-ta* by *-va* after the root *pac*, and 8.2.30 requires substitution of the root-final *-c* by *-k* when an obstruent follows. If *-ta* → *-va* applied first, it would bleed *-c* → *-k*, yielding the wrong form **pacva*. In order to ensure that the suffix change ‘does not count’ with respect to *-c* → *-k*, it is placed *after* it in the *Tripādī*. It is thereby *asiddha* with respect to it, i.e. it fails to bleed it. In this way, both types of opacity can be dealt with by designating a rule as *asiddha*.

Other ways to defeat the *siddha*-principle will be discussed below.

Let us note here that the *siddha*-principle actually extends beyond the ordering relations of feeding and bleeding. The following example serves to illustrate the point. In the derivation of *susyūṣati* ‘wants to sew’, the input *siv+sa+ti* (underlying *ṣivU+saN+tiP*) is subject both to replacement of *v* by *ū* (by 6.4.19 **chvoḥ śūḍ anunāsike ca**) and to reduplication (copying of the root by 6.1.9 **sanyañoḥ**). The *siddha*-principle tells us (correctly) that the the replacement of *v* by *ū* takes effect “before” the copying of the root: *siv+sa+ti* → *syū+sa+ti* → *syū+syū+sa+ti* (→ *suṣyūṣati* by other rules). The reverse procedure would have resulted in the wrong form: *siv+sa+ti* → *siv+siv+sa+ti* (→ **siṣyūṣati* by other rules). There is no question of feeding or bleeding here, but the order of application is transparent whereas the reverse order would be opaque. The reader may wish to check that (38) has the correct effect. Reduplication is done on the base where *v* has been replaced by *ū*, and conversely (but irrelevantly, in this case) the replacement of *v* by *ū* is done on the reduplicated form. This results in the correct output.

2 Syntax

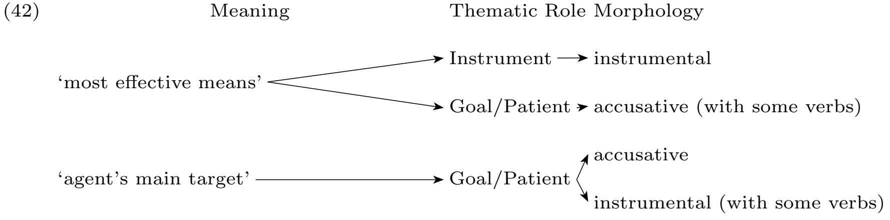
2.1 Pāṇini’s Treatment of Clausal Syntax

Pāṇini accounts for sentence structure by a set of grammatical categories which allow syntactic relationship to be represented as identity at the appropriate level of abstraction. Unlike the phonology and morphology of the *Aṣṭādhyāyī*, the syntax is strictly structure-building. There is no deletion in the syntax, and no movement. Passive sentences are not derived from actives, and nominalizations are not derived from sentences. They are in fact generated in parallel by the same rules in a way which permits their structural parallelism to be captured to the fullest extent.

The pivotal syntactic categories are the *kārakas*. *Kārakas* are roles, or functions, assigned to nominal expressions in relation to a verbal root. They are systematically related to semantic categories, but the correspondence is not one-to-one. One *kāraka* can correspond to several semantic relations and one semantic relation can correspond to several *kārakas*. These correspondences are duly

stated in the grammar. *Kārakas* in turn are the categories in terms of which the assignment of case and other morphological elements is formulated.

(42)



The grammar does not specify any constraints on the order of words in a sentence. This is not surprising since Sanskrit is a "free word order language", even though it is clear that not every permutation of words in a sentence is equally felicitous and that there do exist constraints on scrambling, even though they are poorly understood. For example, the constituents of non-finite complements and adjuncts may not be freely scrambled with their containing clauses. Gérard Huet (p.c.) points out that word order seems to be important for quantification. Even more regrettable from the viewpoint of a modern syntactician is that the grammar is silent on the conditions under which anaphoric elements are bound.

Control relations, such as the determination of the understood agent of infinitives and participles, are stated by means of coreference conditions. There is no question of deletion, or of designated null elements as in many modern tratments of these constructions.

Pāṇini does, however, assume a process of ellipsis, according to which words which are obvious from the context can be omitted. Although ellipsis is an extragrammatical process which is not stated as a rule of grammar, Pāṇini has carefully considered the consequences of this process for his grammatical system and explicitly taken account of them in several syntactic rules of his system.

2.2 Linking and Kārakas

Pāṇini's grammar represents a sentence as a little drama consisting of an action with different participants, which are classified into role types called *kārakas*. There are six of them: Agent (*kartṛ*), Goal (*karman*), Recipient *(saṃpradāna)*, Instrument *(karaṇa)*, Locative *(adhikaraṇa)*, and Source *(apādāna)*.) Some *kārakas* correspond to more than one semantic role and some semantic roles correspond to more than one *kāraka*). Thus, in the grammatical derivation, the *kārakas* mediate between meaning and morphosyntactic surface structure. The grammar provides a set of explicit rules which interprets them semantically and it also contains explicit rules which in turn relate them to various morphosyntactic elements that express them.

Here is the basic paradigm:[2]

[2] To make it easier to follow, I omit *sandhi* here, writing e.g. *pacati odanam* for *pacaty odanam*.

Sentences	**Nominals**
Active:	
kr̥ṣṇa-ḥ pac-a-ti odan-am Krishna-Nom cook-3Sg rice-Acc ‘Krishna cooks rice’	kr̥ṣṇa-ḥ pāc-aka-ḥ odana-sya Krishna-Nom cook-Ag-Nom rice-Gen ‘Krishna (is) a cooker of rice’
kr̥ṣṇa-ḥ svap-iti Krishna-Nom sleep-3Sg ‘Krishna sleeps’	kr̥ṣṇa-ḥ svap-itr̥-∅ Krishna-Nom sleep-Agent-Nom ‘Krishna (is) a sleeper’
Passive:	
kr̥ṣṇ-ena pac-ya-te odana-ḥ Krishna-Ins cook-Pass-3Sg rice-Nom ‘Rice is cooked by Krishna’	kr̥ṣṇ-ena pak-va-ḥ odana-ḥ Krishna-Ins cook-PP-Nom rice-Nom ‘Rice cooked by Krishna’
Stative:	
	kr̥ṣṇ-ena pak-ti-ḥ odana-sya Krishna-Ins cook-Proc-Nom rice-Gen ‘Krishna’s cooking (of rice)’
kr̥ṣṇ-ena sup-ya-te Krishna-Ins sleep-Pass-3Sg ‘Krishna sleeps’ (impers. pass.)	kr̥ṣṇa-sya svap-na-ḥ Krishna-Gen sleep-Proc-Nom ‘Krishna’s sleep’

What is the relation between the active, passive, and stative constructions? And what is the relation between the structure of sentences and the structure of nominals? Pāṇini’s solution is non-derivational on both counts. It introduces a set of grammatical categories in terms of which the related structures can be represented as identical at the appropriate level of abstraction. Actives and passives, sentences and nominals, are alternative realizations of the same underlying relational structure; their derivation is just parallel but — aside from morphological details — identical. In this respect Pāṇini’s grammar represents a *pure form of lexicalism.*

The crucial categories used in this abstract level of representation are the *kārakas.* The basic principles governing the relation between *kārakas* and morphosyntactic surface structure are:

(43) a. Every *kāraka* must be “expressed” (*abhihita*) by a morphological element.
b. No *kāraka* can be expressed by more than one morphological element.
c. Every morphological element must express something.

“Expression” (*abhidhāna*) is thus a three-place relation, so this is a *licensing* approach akin to some modern “linking” theories of clause structure, rather than simply a mapping of two levels. It was, in fact, the inspiration for the first modern formulation of linking theory (Ostler 1979).

Formally, the one-to-one relation between roles and their morphological expression is given by the heading (44), which remains in force until the end of 2.3.

(44) 2.3.1 अनभिहिते
anabhihite
unexpressed-Loc
‘if not (already) expressed’

The derivation of a sentence proceeds as follows. The lexicon associated with the grammar contains a list of verb roots (*dhātu*) and a list of nominal stems that are considered to be underived (*prātipadika*). The derivation is initiated by selecting items from the lexicon and deciding upon a semantic relation between them. The verb's tense is chosen on the basis of the intended time reference (present tense for ongoing time, in these examples, though the full system is quite complex), and grammatical number is assigned to nominal expressions on the basis of how many things they are intended to refer to (in the present example, singular). The participants are assigned their role types; in this case, Agent (*kartṛ*) and Goal (*karman*). Their semantic correlates in these examples are straightforward.

The functions specified for the morphological elements include the following:

(45) a. Instrumental case expresses the Agent and Instrument roles.
b. Accusative case expresses the Goal/Patient role.
c. Active verb endings express the Agent role.
d. Passive verb endings express the Goal/Patient role if the verb has one, otherwise the Process (*bhāva*).
e. Nominalizing suffixes express the Agent (e.g. *-aka*), the Goal/Patient (e.g. the Passive Participle), or the Process (e.g. *-ti*).

(46) 1.4.49 कर्तुरीप्सिततमं कर्म
kartur īpsitatamaṃ karma
agent-Gen most-aimed-at-Nom *karman*
'the primary goal of the *kartṛ* 'agent' is called *karman*'.

The word *īpsitatama* is the superlative of *īpsita* (cf. 1.4.26, 36), which is the past participle of *īps*, the desiderative of *āp* 'attain', so it means 'primary goal'.

The basic expression of *karman* is either the accusative case or the finite verb endings (*la*):

(47) 2.3.2 कर्मणि द्वितीया
karmaṇi dvitīyā
karman-Loc second-Nom
'the second (triplet of case endings) expresses *karman*'.

(48) 3.4.69 लः कर्मणि च भावे चाकर्मकेभ्यः
laḥ karmaṇi ca bhāve cākarmakebhyaḥ (67 *kartari***)**
la-Nom *karman*-Loc and process-Loc and intransitive-AblPl
'*la* expresses *kartṛ*, *karman*, or, after (roots) having no *karman* (intransitives), process.'

Depending on which option is chosen in [48], active or passive sentences are derived.

(49) 1.4.54 स्वतन्त्रः कर्ता
svatantraḥ kartā
independent-Nom agent-Nom
'the independent one is called *kartṛ*.'

The basic expression of *kartṛ* is either instrumental case by [8], or verb inflection by [48]. If the option of expressing the *kartṛ* role by the verb endings is chosen, the *kartṛ* itself is inflected in the nominative case by [50]:[3]

(50) 2.3.46 प्रातिपदिकार्थलिङ्गपरिमाणवचनमात्रे प्रथमा
prātipadikārthaliṅgaparimāṇavacanamātre prathamā
stem-meaning-gender-number-only-Loc first-Nom
'the first case expresses only the gender and number of what the word's stem denotes'

E.g. *devadattaḥ pacati* 'D. cooks', *sthālī pacati* 'the pot cooks', *agniḥ pacati* 'the fire cooks'. Kātyāyana and Patañjali observe that the *kartṛ's* independence is a relative matter and has to do with how the speaker wants to present a situation. Thus, in *sthālī pacati* there is presumably a human agent of cooking at work but since he is not mentioned in the sentence the pot comes to be the independent agent from a grammatical point of view. In *pācayaty odanaṃ devadatto yajñadattena* 'D. makes Y. cook rice', Yajñadatta is independent relative to the act of cooking even though he is in turn prompted by a causer (*hetu*), see [51]:

(51) 1.4.55 तत्प्रयोजको हेतुश्च
tatprayojako hetuś ca
that-prompter-Nom *hetu*-Nom and
'the prompter of that (viz. of the *kartṛ*) is called *kartṛ* and also *hetu* 'cause'.

We will diagram the way roles are expressed by morphology by means of lines linking the verb with the nominals bearing its roles. The morphological element that "expresses" the role is boxed, and subscripted with the name of the role. For example, in [52] the connection line on top shows that the verb inflection *-ti* expresses that *Krishna* is the Agent of *pacati* "cooks", and the one below shows that the accusative ending *-am* expresses that *odana* "rice" is the Goal/Patient of *pacati* "cooks".

(52) kṛṣṇaḥ paca[ti]$_{AGENT}$ odan[am]$_{GOAL}$ 'Krishna cooks rice'

The three mutually exclusive choices available for verb endings (see [45c,d,e]) give rise to respectively the active, passive, and stative sentences and nominals.

Suppose we choose the first option, of having them express the Agent. In that case, they will be spelled out as the appropriate ending of the *active* voice, which will ultimately yield *pacati.* The spell-out rules for *kārakas* then assign the correct case pattern as follows. By the leading principle (43) that each *kāraka* must be expressed, and no *kāraka* may be expressed more than once, since Krishna's Agent role is already expressed by the present tense, this role may not be expressed by an instrumental case on the noun. Krishna is therefore assigned nominative case, which does not express any thematic role. And since the present

[3] This is a non-traditional interpretation but it is the one accepted by most modern scholars.

tense expresses the Agent role, it cannot also express the Goal function of *odana* "rice". Since this role *must* be expressed, an accusative case ending on the nominal is therefore obligatory.

The passive is generated if we instead choose the option of having the verb morphology express the Goal role of *odana* "rice". In that case, *odana* must get nominative case (for if the accusative were chosen, the role would be expressed twice, which is forbidden by (43b)). The Agent function of *kṛṣṇa* must now be expressed by the instrumental ending, which is *-ena*:

(53) kṛṣṇ[ena]$_{AGENT}$ pacya[te]$_{GOAL}$ odanaḥ 'Rice is cooked by Krishna'

The statives are generated by choosing the third option, of having the tense express, not one of the verb's roles, but the verbal "Process" itself. The Agent must then be expressed by the instrumental case ending. For finite verbs, this structure is not available if the verb is transitive (has a Goal).

(54) kṛṣṇ[ena]$_{AGENT}$ supya[te]$_{PROCESS}$ 'Krishna sleeps' (impers. pass.)

The analysis of the corresponding active, passive, and stative nominalizations is entirely parallel to that of the sentences:

(55) kṛṣṇaḥ pāc[aka]$_{AGENT}$ odana[sya]$_{GOAL}$ 'Krishna (is) a cooker of rice'

(56) kṛṣṇ[ena]$_{AGENT}$ pak[va]$_{GOAL}$ odanaḥ 'Rice (is) cooked by Krishna'

(57) kṛṣṇ[ena]$_{AGENT}$ pak[ti]$_{PROCESS}$-ḥ odana[sya]$_{GOAL}$ 'The cooking of rice by Krishna'

(58) kṛṣṇa[sya]$_{AGENT}$ svap[na]$_{PROCESS}$-ḥ 'Krishna's sleep'

2.3 Logical and Grammatical Subject

The concepts "subject" and "object" as such play no role whatsoever in Pāṇini's grammar. We have already seen that there is nothing like a rule that says "the subject is assigned nominative case", or "the object gets accusative case". Then what about the other phenomena which Western grammatical descriptions analyse by means of those grammatical relations, specifically: *(1)* subject-verb agreement, *(2)* anaphora, *(3)* control of PRO. How does Pāṇini cope with them without invoking the subject relation?

2.3.1 Agreement

From the perspective of Western grammar, the verb agrees with the grammatical subject. The system of licensing already described provides an ingenious and simple formulation, and better descriptive coverage. The agreement pattern in active, passive, and stative sentences is shown in [59]:

(59) a. aham pac-ā-mi odana-m
I-Nom cook-1Sg rice-Acc
'I cook rice'

b. aham svap-imi
I-Nom sleep-1Sg
'I sleep' (active)

c. mayā pac-ya-te odana-ḥ
I-Instr cook-Pass-3Sg rice-Nom
'Rice is cooked by me'

d. mayā sup-ya-te
I-Instr sleep-Pass-3Sg
'I sleep' (impersonal passive)

The verb agrees with the 1Sg pronoun in [59a,b], but not in [59c,d]. Why? Recall that the finite verb ending "expresses" one of three things: *(1)* the Agent, *(2)* the Goal, or *(3)* the Process. So the relational structures of [59] are:

(60) a. aham pacā [my]$_{AGENT}$ odan [am]$_{GOAL}$ 'I cook rice'

b. aham svap [imi]$_{AGENT}$ 'I sleep'

c. [mayā]$_{AGENT}$ pacya [te]$_{GOAL}$ odanaḥ 'Rice is being cooked by me'

d. [mayā]$_{AGENT}$ supya [te]$_{PROCESS}$ 'I sleep' (impersonal passive)

and the agreement rule we need is:

(61) If the verb ending expresses the role of a first or second person pronoun, it is first or second person, otherwise it is third person.

The "elsewhere case" (third person) arises both when the ending expresses the *kāraka* of a third person nominal, as in [60c], and when it expresses no *kāraka* at all, but rather "Process", as in [60b,d]. The verb and the nominal so linked are designated by Pāṇini as *samānādhikaraṇa*, "having the same substratum" (let's translate it as *coindexed*, for it isn't quite the same thing as "coreferential"). It is

the same relation as that which holds between the head and modifiers in a noun phrase. Thus, subject-verb person agreement and adjective-noun gender/number concord are in this treatment manifestations of the same semantic relationship.

Independent motivation for the coindexing treatment comes from the agreement of conjoined subjects. A conjunct nominal of the form 1P+3P gets 1P agreement, a conjunct nominal of the form 2P+3P gets 2P agreement:

(62) aham kṛṣṇa-ś ca pacā-vaḥ
I-Nom Krishna-Nom and cook-1Dual
I and Krishna are cooking

This follows without further stipulation from the rules already given.

Another way in which Pāṇini's coindexing account of agreement is superior to a subject-based account is seen in relative clauses like [63].

(63) paśya-ti mām ya-ḥ pacā-mi
see-3Sg me-Acc who-Nom cook-1Sg
'He sees me, who am cooking'

The "subject" is the third person relative pronoun *ya-* "who", but, as rule [61] predicts, the verb in fact agrees with the *head* of the relative clause, in this case with *aham* "I". Any rule specifying agreement in terms of subjects would have to be supplemented to get first or second person agreement here. Pāṇini's rule, however, covers this situation directly. In [63], the verb is coindexed with both *aham* "I" (first person) and with *yaḥ* "who" (third person), and first person verb agreement is correctly enforced as before.

Since Sanskrit is a typical "*pro*-drop language" the pronouns which trigger first and second person verb agreement are not necessarily expressed. Indeed, they are usually absent. Pāṇini's grammar simply assumes that words can be freely omitted in sentences when they are evident from the meaning or context. This was considered a phenomenon which falls outside the domain of sentence grammar, so there are no rules of ellipsis in the grammar. Still, we can be sure that Pāṇini considered ellipsis a rule-governed discourse process because some rules of grammar that he does formulate make reference to the results of ellipsis, for reasons that have to do with his general theory of rule interaction. This says that if rule A has the effect of bleeding rule B (eliminating inputs to it), A has priority over B in the order of application; hence ellipsis should take place before any rule that refers to the ellipsed elements — in other words, ellipsed elements should have no syntactic effects. In cases where this convention does *not* yield the right results, Pāṇini explicitly arranges for the ellipsed elements to be "visible" by means of special conditions on rules. This is the procedure he follows to prevent *pro*-drop from bleeding agreement. He complicates his agreement rule to say that pronouns trigger agreement even when they are not overtly expressed. In this way, agreement is entirely unaffected by whether the triggering pronoun is overtly present or subjected to ellipsis.

To ensure that person agreement applies properly even when pronouns are dropped, the agreement rules (64), (65) must stipulate that agreement takes place even when the pronoun is present just in the input *(sthāniny api).*

(64) 1.4.105 युष्मद्युपपदे समानाधिकरणे स्थानिन्यपि मध्यमः
yuṣmady upapade samānādhikaraṇe sthāniny api
yuṣmad-Loc adjunct-Loc coindexation-Loc input-Loc even
madhyamaḥ
middle
'the second set of person endings are added in coindexation with the stem *yuṣmad* 'you', even if underlying'

(65) 1.4.107 अस्मद्युत्तमः
asmady uttamaḥ (105 upapade samānādhikaraṇe sthāniny
asmad-Loc last
api)

'the last set of person endings (= first person) are added in coindexation with the stem *asmad* 'I', even if underlying'

(66) 1.4.108 शेषे प्रथमः
śeṣe prathamaḥ
rest-Loc first
'elsewhere the first set of person endings (= third person) are added'

Thus, *kārakas* make possible an ingenious treatment of verb agreement which not only gets around the potential problem which the lack of the concept "subject" would seem to cause, but actually explains a set of cases which are problematic for accounts of agreement that rely on such a concept.

2.3.2 Anaphora

The second important area where the subject relation is invoked in Western grammar is anaphora (Binding Theory). For example, the antecedent of a reflexive pronoun in many languages must be a grammatical subject. As it happens, the antecedent of a reflexive pronoun in Sanskrit is normally the *logical* subject — for example, the Agent phrase in a passive. This generalization is inexpressible in any binding theory which relies on the notion "grammatical subject" (however defined), but it is easy in Pāṇini's system. The antecedent is just the *kartṛ*. Again, the absence of grammatical relations, far from being a liability of Pāṇinian grammar, turns out to be consistent with the right generalizations about Sanskrit.

2.3.3 Control and Ellipsis

The third area where grammatical relations might play a role is "control" of the PRO subject of nonfinite verbs (infinitives and absolutives). In Sanskrit, this depends on the *kārakas* in an absolutely essential way. Contrary to English, where PRO subjects are coreferential with the containing clause's subject or object, in Sanskrit they are coreferential with its Agent or Goal. In particular, if the main clause is passive, the controller of an adverbial participial clause is the grammatical subject in English, and the Agent (its "logical subject") in Sanskrit. For example, the literal English translation of our example sentence [2]

entails that the rice arrived in the village, in Sanskrit it means that Āśvapata arrived in the village. Pāṇini states the appropriate control principle at the level of thematic roles, as a condition requiring that the absolutive must have the same Agent as its governing verb.

(67) 3.4.21 समानकर्तृकयोः पूर्वकाले
samānakartr̥kayoḥ pūrvakāle (19 ktvā) (3.1.91 dhātoḥ) (3.1.1-2
same-agent-AblPl prior-Loc
pratyayaḥ. . .)

'When two events have the same agent, *Ktvā* is suffixed to the root which denotes the prior event.'

This accounts correctly for the interpretation of the absolutive in sentences such as (2). It also predicts correctly that Sanskrit has no passive absolutives (i.e. no analogs to English *having been eaten*).

So far so good. But when we turn to infinitive purpose clauses, a question arises about how the licensing mechanism works in these control structures. The infinitive of purpose is introduced by rule

(68) 3.3.10तुमुन्ण्वुलौ क्रियायां क्रियार्थायाम्
tumunṇvulau kriyāyāṃ kriyārthāyām
tumUN-ṆvuL-NomDu action-Loc action-purpose-Loc
'(the *kr̥t* affixes) *tumUN* and *ṆvuL* are added (after a root) denoting an action which is the purpose of another action'

Thus, the infinitive *-tum* seems to express the Agent role (in virtue of being a *kr̥t* suffix), and purpose. We seem to face the problem that the theory apparently predicts that sentences like [69a,b] are ungrammatical, and [69c] grammatical, and the facts are exactly the other way around:

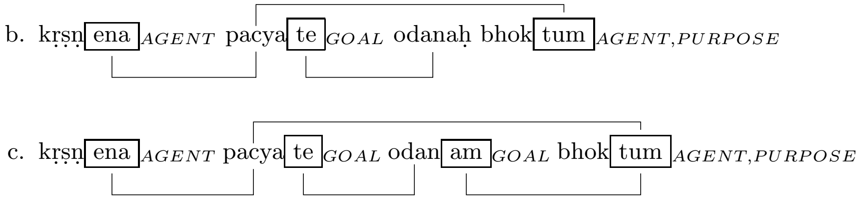
(69) a. kr̥ṣṇa-ḥ paca-ty odana-m bhok-tum
Krishna-Nom cook-3Sg rice-Acc eat-Inf
'Krishna cooks rice to eat'

b. kr̥ṣṇ-ena pac-ya-te odana-ḥ bhok-tum
Krishna-Instr cook-Pass-3Sg rice-Nom eat-Inf
'Rice is cooked by Krishna to eat'

c. *kr̥ṣṇ-ena pac-ya-te odana-m bhok-tum
Krishna-Instr cook-Pass-3Sg rice-Acc eat-Inf
'Rice is cooked by Krishna to eat'

The reason the system seems to predict wrongly that [70a] and [70b] is ungrammatical is that the Goal role that *odana* "rice" bears in relation to *bhoktum* "to eat" is not expressed, and the reason it seems to predict that in [70c] is grammatical is that all roles are apparently properly expressed:

(70) a. kr̥ṣṇa-ḥ paca[ti]$_{AGENT}$ odan[am]$_{GOAL}$ bhok[tum]$_{AGENT,PURPOSE}$

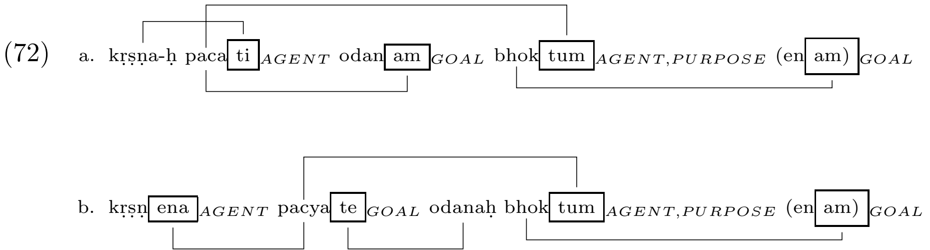


The solution to this problem is a constraint, apparently presupposed by Pāṇini, to the effect that:

(71) A nominal cannot have a role with respect to more than one verb. (The "Th-criterion".)

So, since *odana* is the Goal of *pacati* in the sentences of [70], it can't also also bear a role with respect to the subordinate infinitive *bhoktum*. Accordingly, [70c] is ungrammatical because the accusative case of *odanam* has no function (does not express any role). In the grammatical sentences [70a,b], all morphological elements are licensed, so the sentences are OK.

But then what expresses the relation between *odana* and *bhoktum* in those sentences? The answer is: nothing. There is in fact no direct relation between them. The Goal of the infinitive *bhoktum* is not *odana* itself, but an ellipsed pronoun in the subordinate clause that anaphorically refers to *odana*.

(72)


This analysis correctly predicts the possibility of sentences in which the main verb's object and the infinitive's object are distinct:

(73) a.

kṛṣṇa-ḥ	kāṣṭhāni	bhinat-ti	odana-m	pak-tum
Krishna-Nom	firewood-pieces	split-3Sg	rice-Acc	cook-Inf

'Krishna splits firewood in order to cook rice'

b.

kṛṣṇ-ena	kāṣṭhāni	bhid-ya-nte	odana-m	pak-tum
Krishna-Instr	firewood-pieces	split-Pass-3Pl	rice-Acc	cook-Inf

'Firewood is split by Krishna in order to cook rice'

The infinitive constructions in question furnish another interesting example of an explicit reference to ellipsis in Pāṇini's rules. Consider the range of elliptic variants of a sentence such as

(74)

aham	edh-ān	āhar-tum	gacchā-mi
I-Nom	firewood-Acc	fetch-Inf	go-1Sg

'I'm going to fetch firewood'

As before, the pronoun *aham* can be freely omitted without consequences for the syntax of the sentence. The same is true, with one exception, for the other words in the sentence, in the appropriate contexts:

(75) a. *edhān āhartum* ('Where are you going?') 'To fetch firewood'.
b. *āhartum gacchāmi* ('What about the firewood?') 'I'm going to fetch (it)'.
c. *edhān* ('What are you going to fetch?') 'Firewood'.
d. *āhartum* ('What are you going to do to the firewood?') 'Fetch (it)'.
e. *gacchāmi* ('Are you fetching the firewood?') 'I'm going to'.

However, one of the logically possible patterns of deletion, [76a], is ungrammatical, just as its English counterpart. But if we change the case of the noun into the dative, the sentence is OK again!

(76) a. **edhān gacchāmi* 'I'm going firewood'.
b. *edhebhyo* (Dative) *gacchāmi* 'I'm going for firewood'.

To account for the ungrammaticality of [76a] and for the grammaticality of [76b] Pāṇini adds a special rule of case assignment to his grammar.

(77) 2.3.14 क्रियार्थोपपदस्य च कर्मणि स्थानिनः
kriyārthopapadasya ca karmaṇi sthāninaḥ (13
action-meaning-complement-Gen and *karman*-Loc substitute-Gen
caturthī)

'The fourth set of case endings (dative) expresses the *karman* of a substituted [in effect, deleted] verb which denotes an action whose purpose is another action.' which is

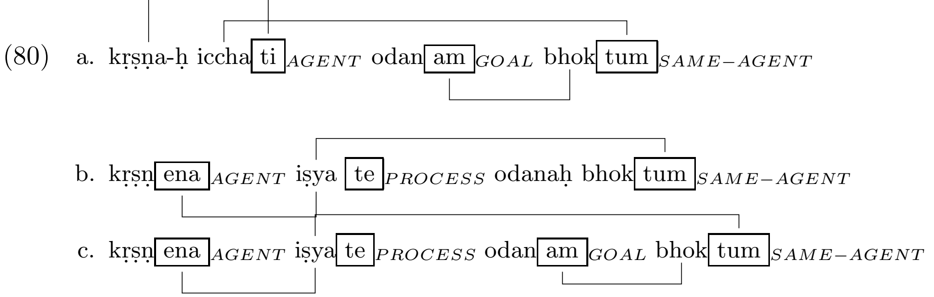
The verb deleted in [76a] is *āhartum* "to fetch"; it is construed with *gacchāmi* by Pāṇini's rule for purpose complements (3.3.10). Therefore, 2.3.14 applies to yield [76b]. In accord with the *siddha*-principle (38), the deleted verb would be invisible were it not for this rule. In accord with the blocking principle, (77) automatically blocks the general rule which states that accusative case expresses a Goal.

Returning to control, in a superficially similar construction with a class of "equi" verbs, such as *icchati* "want", the pattern of grammaticality judgments in the passive is the opposite of that in [70b,c]:

(78) a. kṛṣṇa-ḥ iccha-ti odana-m bhok-tum
Krishna-Nom want-3Sg rice-Acc eat-Inf
'Krishna wants to eat rice'
b. *kṛṣṇ-ena iṣ-ya-te odana-ḥ bhok-tum
Krishna-Instr want-Pass-3Sg rice-Nom eat-Inf
'Rice is wanted by Krishna to eat' (contrast the grammatical [70b])
c. kṛṣṇ-ena iṣ-ya-te odana-m bhok-tum
Krishna-Instr want-Pass-3Sg rice-Acc eat-Inf
'Rice is wanted by Krishna to eat' (contrast the ungrammatical [70c])

For these verbs, Pāṇini provides a specific control rule.

(79) 3.3.158 समानकर्तृकेषु तुमुन्
samānakartṛkeṣu tumun **(157 icchārtheṣu)**
same-agent-ed-LocPl tumUN-Nom
'*tumUn* is added after a root that is construed with a root denoting 'want' that has the same agent'



Note that in [80b] *-te* (in its passive function) cannot link *odanaḥ*, because of principle [71].

With yet a third class, "raising"-type verbs such as *śak* "can", the pattern in the passive is once more reversed:

(81) a. kṛṣṇa-ḥ śakno-ti odana-m bhok-tum
Krishna-Nom can-3Sg rice-Acc eat-Inf
'Krishna can eat rice'
b. kṛṣṇ-ena śak-ya-te odana-ḥ bhok-tum
Krishna-Instr can-Pass-3Sg rice-Nom eat-Inf
(passive of [a])
c. *kṛṣṇ-ena śak-ya-te odana-m bhok-tum
Krishna-Instr can-Pass-3Sg rice-Acc eat-Inf
(passive of [a])

Pāṇini explicitly provides for this class too by a special rule which stipulates that the infinitive suffixes, such as *tum*, in connection with roots of the *śak* class, express *dhātusambandha* "verb union" (Joshi 1971).

(82) 3.4.1 धातुसंबन्धे प्रत्ययाः
dhātusaṃbandhe pratyayāḥ
root-connection-Loc suffix-PlNom
'(words ending in the following) suffixes are connected to verbal roots'

In virtue of this verb union process, combinations like *bhoktum śaknoti* "can eat" are treated exactly like a single predicate for purposes of the grammar's licensing constraints. The data in [81] then follows:

(83) a. kṛṣṇa-ḥ śakno [ti]AGENT odan [am]GOAL bhok [tum]DHĀTUSAMBANDHA

b. kṛṣṇ [ena]AGENT śakya [te]GOAL odanaḥ bhok [tum]DHĀTUSAMBANDHA

c. kṛṣṇ [ena]AGENT śakya [te]GOAL odan [am]GOAL bhok [tum]DHĀTUSAMBANDHA

Finally, participial constructions with object control require a special rule:

(84) 3.2.124 लटः शतृशानचावप्रथमासमानाधिकरणे
laṭaḥ śatṛśānacāv aprathamāsamānādhikaraṇe
lAṬ-Gen ŚátṚ-ŚānaC-NonDu non-Nominative-coindexed-Loc
'When present tense *lAṬ* is not coindexed with a nominative, it is replaced by the participle endings *-āna, -a(n)t*'.

This rule obligatorily turns [85a] into [85b], and thus at one stroke accounts for all the data — for the ungrammaticality of [85a], which could otherwise be parsed by the system, for the grammaticality of [85b], and for the ungrammaticality of the corresponding passive [85c], which violates [84].

(85) a. *kṛṣṇa-ḥ paśya-ti Yajñadatt-am gaccha-ti
Krishna-Nom see-3Sg Yajñadatta-Acc go-3Sg
'Krishna sees Yajnadatta go'

b. kṛṣṇa-ḥ paśya-ty Yajñadatt-am gacch-ant-am
Krishna-Nom see-3Sg Yajñadatta-Acc go-Part-Acc
'Krishna sees Yajnadatta goes'

c. *kṛṣṇ-ena dṛś-ya-te Yajñadatta-ḥ gacch-an
Krishna-Instr see-Pass-3Sg Yajñadatta -Nom go-Part-Nom
'Yajnadatta is seen go by Krishna'

2.4 Non-thematic Objects

Any linking theory must deal with the fact that verbs can have objects which are not thematically related to them. I would like to propose that this is intended to be covered by an additional rule which assigns the *karman* role:

(86) 1.4.50 तथायुक्तं चानिप्सीतम्
tathāyuktaṃ cānīpsitam (49 kartuḥ, karma)
so-connected and non-goal-Nom
'also that which is likewise connected, even if not a goal of the agent, is called *karman*'.

Its wording is obscure and the interpretation is disputed. Traditionally it is taken to cover 'non-desired' objects, e.g. *viṣaṃ bhakṣayati* 'he eats poison'. More recently the proposal is gaining ground that its purpose is to account for the cases where double objects are permitted: e.g. *gāṃ payo dogdhi* 'he milks the cow (of)

milk’, *pauravaṃ gāṃ bhikṣate* ‘he begs Paurava for a cow’, *mānavakaṃ dharmaṃ brūte* ‘he teaches the boy duty’, *gargān śataṃ daṇḍayati* ‘he fines the Gargas a hundred’, *ajāṃ grāmaṃ nayati* ‘he leads the goat to the village’. Joshi suggests that *tathāyuktam* specifies that the second *karman* allowed by this rule should be connected to the main *karman* allowed by the preceding rule. However, this should have been expressed as **tadyuktam.* Kiparsky 1982, 41 proposes that this is in effect a transderivational rule, which sanctions the assignment of *karman* to an argument which does not satisfy the definition of [46] **kartur īpsitatamaṃ karma** provided that there is another sentence in which it does function as a *karman* by rule [46]. Thus, *gāṃ payo dogdhi* would be licenced by the existence of both *gāṃ dogdhi* ‘he milks the cow’ and *payo dogdhi* ‘he milks milk’, but in the double-object construction only one of them is *īpsita.* Ungrammatical sentences such as **gāṃ dogdhi kumbham* ‘he milks the cow the pot’ would be excluded because the second *karman* is not *tathāyukta* (cf. **kumbham dogdhi*). Ungrammatical sentences such as **gāṃ dogdhy ajām* ‘he milks the cow the goat’ would be excluded because they would have to have two parallel *īpsita karmans.*

My suggestion is that the same rule extends to non-thematic objects in the “conjunct participle” construction, which are otherwise not covered by the grammar. In this construction, a participial modifier constitutes the semantic predicate of its head noun, combining with it into the functional equivalent of an argument or adjunct clause. The head can be an object, as in (87a,b), or a passivized subject, as in (87c):

(87) a. **taṃ mantriṇā hataṃ śrutvā nyavedayan ...**
him-Acc minister-Instr kill-PPP-Acc hear-Abs inform-Impf-3Pl ...
‘after hearing that the king had been killed by his minister, they informed ... ’ ($\not\supset$ ‘after hearing the king’) (Mbh. 3.283.4)

b. **taṃ vaí pakṣapuchávantam eva sántam ná**
him-Acc Part wing-tailed-Acc just being-Acc not
pakṣapuchávantam iva paśyanti
wing-tailed-Acc as-if see-3Pl
‘although he in fact has wings and a tail, people do not see him as having wings and a tail’ ($\not\supset$ ‘people do not see him’) (ŚB 7.1.1.20)

c. **havyavāhanaḥ śrūyate nigr̥hīto vai purastāt**
fire-Nom hear-Pass3Sg caught-Nom Prt once
pāradārikaḥ
adulterer-Nom
‘fire is heard (said) to have been once caught as an adulterer’ ($\not\supset$ ‘fire is heard’) (*Mbh* 2.28.17)

The objects *tam* in (87a) and in (87b), and the passivized subject *havyavāhanaḥ* in (87c), do not bear a thematic relation to the main verb that governs them; their thematic role comes from the conjunct participle. Thus, they do not satisfy the definition of the *karman* relation in (46). To receive accusative case, and to be passivized in accord with the grammar, they must however have the status of *karman.* My proposal is that these sentences are *transderivationally* sanctioned

by (86) through analogy to parallel sentences where the corresponding argument is in fact legitimately an object, viz. *taṃ śrutvā* 'having heard him', *taṃ paśyanti* 'they see him', *havyavāhanaḥ śrūyate* 'the fire is heard.'

A problem is that the participle constitutes an anaphoric domain, in the sense that its implicit subject can (anaphorically) control the null subject of embedded nonfinite clauses:

(88) **māṃ tu dr̥ṣṭvā pradhāvantam anīkaṃ saṃpraharṣitum**
me-Acc Part see-Abs rushing-Acc front-Acc cheer-Inf
tyajantu harayas trāsaṃ
abandon-Imp3Pl monkey-Pl fear-Acc
'when they see me rush to the front to restore morale, may the monkeys lose their fear' ((*R.* 6.360.37)

Here the agent of *saṃpraharṣitum* is *māṃ* 'me' and not *harayas* 'the monkeys'. Therefore, it is not just a *karman* but also a *kartr̥*.

The third rule that introduces the *karman* role is:

(89) 1.4.51 अकथितं च
akathitaṃ ca (49 kartur īpsitatamaṃ karma)
unstated-Nom also
'Also an unexpressed primary goal of the agent is *karman*'.

I think *akathita* should be understood literally as *asaṃkīrtita* 'not mentioned, left out' or *avivakṣita* 'not intended to be expressed' *in the sentence* and that the rule takes care of "object pro-drop".[4] It specifies that ellipsed *karmans* are to count as *karmans* for purposes of the rules of grammar. About a dozen rules of the grammar distinguish between verbs with and without a *karman* (*akarmaka* vs. *sakarmaka*, roughly intransitive vs. transitive). It is crucial that transitive verbs count as *sakarmaka* even when their object is not overtly expressed. For example, rule 1.4.52 states that the agent of *akarmaka* verbs is a *karman* in the causative, e.g. *devadattam āsayati* 'he makes D. sit down'. But verbs with implicit unexpressed *karmans* are still *sakarmaka*, e.g. *dohayaty aśūdreṇa* (not **aśūdram*) 'he makes a non-*śūdra* milk (it)'. To get this it was necessary to frame the present rule because the *siddha*-principle would otherwise cause ellipsed elements to be invisible (Kiparsky 1982, 41-44).

2.5 Semantic Competition among *kārakas*

The rules assigning *kāraka* designations are headed by (90), (91), which ensure that any given argument gets only one role (semantic blocking again).

(90) 1.4.1 आ कडारादेका संज्ञा
ā kaḍārād ekā saṃjñā
'up to *kaḍāra*-Abl one term

[4] Tradition takes it to mean that whatever is not mentioned in the previous rules is *karman*, but is unable to provide a satisfactory interpretation on this basis.

'up to [the first occurrence of the word] *kaḍāra* (the end of 2.2), (only) one technical term (is to be assigned)'

(91) 1.4.2 विप्रतिषेधे परं कार्यम्

vipratiṣedhe paraṃ kāryam
conflict-Loc last to be applied'
'in case of conflict, the last (rule) is to be applied'

For example, in *dhanuṣā vidhyati* 'he pierces by means of a bow' (i.e. with arrows shot from a bow), *dhanuṣ* 'bow' satisfies both the definition of the source role (*apādāna*), and the definition of the instrument role (*karaṇa*), since it is both the "fixed point" from which the arrows move off and the "means" for launching the arrows.

(92) 1.4.24 ध्रुवमपाये ऽपादानम्

dhruvam apāye 'pādānam
fixed-Nom separation-Loc *apādāna*-Nom
'in a separation, the fixed point is (called) *apādāna* ('source')'

(93) 1.4.42 साधकतमं करणम्

sādhakatamaṃ karaṇam
most-effective-means-Nom *karaṇa*
'the most effective means is (called) *karaṇa* ('instrument')'

As a result of [90], [91] it is only designated as a *karaṇa* by the later rule 1.4.42, so that the grammatical sentence *dhanuṣā vidhyati* 'he pierces with a bow' is derived and the ungrammatical sentence *dhanuṣo vidhyati* 'he pierces from a bow' is not derived.

The heading that introduces the *kāraka* terms is continued through [51] 1.4.55.

(94) 1.4.23 कारके

kārake
role-Loc
'to express a role'

The characterization of the *kāraka* roles raises some interesting questions that semanticists still wrestle with. Let us sample this domain by taking a look at the discussion around the *kāraka* called *apādāna* 'source', introduced by (92): The basic expression for *apādāna* is the ablative case.

(95) 2.3.28 अपादाने पञ्चमी

apādāne pañcamī
source-Loc fifth-Nom
'the fifth case expresses *apādāna*'

E.g. *vanāt* 'from the forest' (see (2)). Kātyāyana notes that in cases such as *aśvāt trastāt (dhāvataḥ) patitaḥ* 'he has fallen from a shying (running) horse', *sārthād gacchato hīnaḥ* 'he has strayed from the moving caravan' the horse and caravan are not fixed. Patañjali argues that this is no problem because they are fixed *in relation to* the object which moves away from them (Joshi & Roodbergen 1975).

The main problem is how to characterize the source relation outside of the domain of physical movement. Subsequent special rules (all with *apādānam* continued from (92)) provide explicitly for this. (96) and (97).

(96) 1.4.25 भीत्रार्थानां भयहेतुः

bhītrārthānāṃ **bhayahetuḥ**
fear-protect-meaning-GenPl fear-cause

'In connection with (roots) meaning 'fear' and 'protect' the cause of fear is (called) *apādāna*'.

E.g. *caurebho bibheti* 'he fears thieves', *caurebhyo rakṣati* 'he protects from thieves', and in derivatives such as *bhīma* (by 3.4.74 *bhīmādayo 'pādāne*). Patañjali retorts that these verbs denote a mental separation and therefore are already covered by Pāṇini's rule. Similarly:

(97) 1.4.26 पराजेरसोढः

parājer **asoḍhaḥ**
overcome-Gen unbearable-Nom

'In connection with *parāji* "to be overcome", that which one cannot endure is (called) *apādāna.*'

E.g. *adhyayanāt parājayate* 'he is tired of studying'. Patañjali says: "A thoughtful person observes: study is a pain, it is difficult to memorize things, and teachers are hard to satisfy. (And so,) having (first) formed a connection (with study) in his mind, he (then) desists (from it). This being so, we can manage by (1.4.24) **dhruvam apāye 'pādānam**." Kātyāyana notes that roots denoting disgust, cessation, and neglect should be specified in the rule: *adharmāj jugupsate* 'he is disgusted by wrong', *dharmān muhyati* 'he neglects *dharma*', for which Patañjali proposes the same account.

Going further, Patañjali proposes that the basic rule (92) can handle examples like *yavebhyo gā vārayati/nivartayati* 'he wards off/turns away the cows from the barley', *kūpād andhaṃ vārayati* 'he keeps the blind man away from the well', *upādhyāyād antardhatte* 'he hides from the teacher', *upādhyāyād adhīte* 'he learns from his teacher', *śṛṅgāc charo jāyate* 'the arrow is made out of horn', *gomayād vṛściko jāyate* 'the scorpion originates from cowdung', *himavato gaṅgā prabhavati* 'the Ganges arises from the Himalayas', so that all the special rules for them can be eliminated.

For more recent insightful discussion of various non-movement source and goal relations see Talmy 1988 and Fong 1997. Fong defines a more abstract notion of source which can be parametrized in different domains.

3 Morphology

3.1 Categories and Word-Formation Processes

Suffixation consists of adding a suffix *(pratyaya)* to a base *(aṅga)*. The rules in chapters 3 - 5 (a section of about 1800 rules) deal with suffixation, and are headed by (10) and (11). A base of affixation is defined as *aṅga* by [98]:

(98) 1.4.13 यस्मात्प्रत्ययविधिस्तदादि प्रत्यये ऽङ्गम्
yasmāt pratyayavidhis tadādi pratyaye 'ṅgam
what-Abl suffix-rule-Nom that-beginning-Nom suffix-Loc base-Nom
'whatever an suffix is appended to, together with anything that follows it before the suffix, is an *aṅga* "base"'

Bases are of three categories:

- *dhātu* '(verbal) root'
- *prātipadika* '(nominal) stem'
- *pada* 'word'

Roots are either basic (defined in [99]) or derived (defined in [100]).

(99) 1.3.1 भूवादयो धातवः
bhūvādayo dhātavaḥ
bhū-beginning-PlNom root-PlNom
'*bhū* etc. (the items listed in the *dhātupāṭha*) are (termed) *dhātu* "root"'

(100) 3.1.32 सनाद्यन्ता धातवः
sanādyantā dhātavaḥ
saN-beginning-ending-PlNom root-PlNom
'items ending in *san* etc. (the suffixes introduced in rules 3.1.5 ff.) are (termed) *dhātu* "root"'.

Nominal stems are also either basic or derived. Basic nominal stems are defined in [101].

(101) 1.2.45 अर्थवदधातुरप्रत्ययः प्रातिपदिकम्
arthavad adhātur apratyayaḥ prātipadikam
meaning-having-Nom non-root-Nom non-suffix-Nom base
'an element which has a meaning and is not a *dhātu* or a *pratyaya* [and does not end in a *pratyaya*], is (termed) *prātipadika*' "base"'.

The definition excludes not only suffixes, but also suffixed items. Words *(padas)*, for example, are not *prātipadikas*, for goodmany reasons. Suffixes, are *prātipadikas*, however, in so far as they fall under [102].

(102) 1.2.46 कृत्तद्धितसमासाश्च
kṛttaddhitasamāsāś ca (45 prātipadikam)
kṛt-taddhita-compound-PlNom and'

'elements ending in *kṛt* or *taddhita* suffixes, and compounds, are (termed) *prātipadika*'.

Rule (102) actually covers the majority of suffixed nominal forms, excepting only finished nouns *(padas)* and feminine stems (although the latter pattern with *prātipadikas* as inputs to inflection and secondary derivation, as provided for by (13) 4.1.1 **ñyāp prātipadikāt**).

A WORD is defined as anything that ends in an inflectional suffix.

(103) 1.4.14 सुप्तिङन्तं पदम्

sup-tiṅ-antam padam

suP-tiṄ-ending-Nom word-Nom

'An element tha ends in *suP* (a case ending) or *tiṄ* (a person/number ending) is (termed) *pada* 'word'.

The definition covers indeclinable words too, for they are all assigned nominal inflectional endings, which are then deleted. Similarly, each member of a compound is a word because it contains a later deleted case ending. For instance, *rājapuruṣa* 'king's servant' is derived from *rājan-Ṅas+puruṣa-sU*, with an internal genitive case ending on the first member. The reason for this procedure is that it simplifies the morphological derivation of compounds and automatically accounts for certain phonological phenomena. For example, in *rājapuruṣa* 'king's servant' the first member *rājan-*, being a word, gets its correct form by an independently motivated phonological rule which deletes word-final -*n*.

How can the wordhood of an indeclinable or of the first member of a compound be due to its deleted case ending, given that the *siddha*-principle says that deleted material is invisible? The reason even deleted case endings confer wordhood is rule [104]:

(104) 1.1.62 प्रत्ययलोपे प्रत्ययलक्षणम्

pratyayalope pratyayalakṣaṇam

suffix-deletion-Loc suffix-effect-Nom

'when a suffix is deleted, the operations triggered by it still apply'

It sets aside the *siddha*-principle, so that words whose case endings are deleted by still count as *padas* for purposes of applying [103].

The following types of word-formation occur:

a. $[\text{Root} + \text{Suffix}]_{\text{Root}}$: desideratives, intensives, causatives.
b. $[\text{Word} + \text{Suffix}]_{\text{Root}}$: denominal verbs.
c. $[\text{Root} + \text{Suffix}]_{\text{Stem}}$: primary *(kṛt)* suffixes.
d. $[\text{Word} + \text{Suffix}]_{\text{Stem}}$: secondary *(taddhita)* suffixes.
e. $[\text{Word} + \text{Word}]_{\text{Stem}}$: compounding.
f. $[\text{Root} + \text{Suffix}]_{\text{Word}}$: verb inflection.
g. $[\text{Stem} + \text{Suffix}]_{\text{Word}}$: noun inflection.

3.2 The "Sup" Endings

Nominal stems (*prātipadikas)* are marked by suffixes for number and case. Sanskrit has three numbers (Singular, Dual, and Plural) and seven cases (Nominative, Accusative, Instrumental, Dative, Ablative, Genitive, Locative). (The vocative is not considered a separate case, but a use of the nominative, even though it does have distinctive endings in the singular.)

The underlying endings are enumerated in rule [105].[5]

(105) 4.1.2

	Singular *(ekavacana)*	Dual *(dvivacana)*	Plural *(bahuvacana)*	
	sU	au	Jas	Nominative *(prathamā)*
	am	auṬ	Śas	Accusative *(dvitīyā)*
	Ṭā	bhyām	bhis	Instrumental *(tṛtīyā)*
	Ṅe	bhyām	bhyas	Dative *(caturthī)*
	ṄasI	bhyām	bhyas	Ablative *(pañcamī)*
	Ṅas	os	ām	Genitive *(ṣaṣṭhī)*
	Ṅi	os	suP	Locative *(saptamī)*

As usual, *pratyāhāras* are formed from this list by combining a listed element with a diacritic to include all the intervening elements in the list, e.g. *suP* 'case ending', *suṬ* 'a "strong" case ending'. The endings are numbered in successive groups of three, with *(prathamā)* "first", *(dvitīyā)* "second", etc. serving as names of the cases. The first ending in each group is given the designation *ekavacana* 'singular', the second *dvivacana* "dual', and the third *bahuvacana* "plural".

The first case and the last three make up the basic format of a Pāṇinian substitution rule. Genitive case marks the item to be replaced, Nominative the replacement, Ablative the left context, and Locative the right context.

The underlying forms in (105) are basically those of the -C stems. In the *-a* declension, most of the singular endings have suppletive alternants. For example, the Instr., Abl., and Gen.Sg. endings are introduced by (106):

(106) 7.1.12 टाङसिङसामिनात्स्याः
ṭāṅasiṅasām inātsyāḥ (9 ataḥ) (6.1.4 aṅgasya)
Ṭā-ṄasI-Ṅas-PlGen *ina-āt-sya*PlNom
After a base ending in short *a*, the case endings *Ṭā, ṄasI, Ṅas* are replaced (respectively) by *ina, āt, sya.*

E.g. Instr.Sg. *Āśvapata+ā* → (106) *Āśvapata+ina* → [26] *Āśvapatena.*

This brings up another important rule which trumps the *siddha*-principle in a special set of cases:

[5] For the reader's convenience the items enumerated in the rule are here arranged into labeled columns and rows, with *sandhi* undone. Actually, of course, it is recited as running text, like the whole grammar, and the labels are assigned by rules as explained directly below.

(107) 1.1.56 स्थानिवदादेशो ऽनल्विधौ
sthānivad ādeśo 'nalvidhau
original-like-Nom substitute-Nom non-sound-rule-Loc'
'a substitute is treated like the original, except with respect to a phonologically conditioned operation'

This rule says that *non-phonological* properties of the input *are* inherited under replacement. Consider the derivation of Dat.Sg. *grāmāya*. After *-a* stems, Dat.Sg. *N̄e* is replaced by *ya* by 7.1.13. The replacement then triggers lengthening by [108].

(108) 7.3.102 सुपि च
supi ca (101 ataḥ) (101 dīrghaḥ, yañi) (6.4.1 aṅgasya)
sUP-Loc and
'The final *a* of a nominal stem is lengthened before a case ending that begins with yaÑ (a glide, a nasal, *jh*, or *bh*).'

But [108] calls for lengthening before a *sup* (case ending), and, being introduced as a replacement, *-ya* obviously does not appear in the list of case endings that is subsumed under the *pratyāhāra sup* in [105]. Then how can *-ya* trigger the desired lengthening? The answer is that it "inherits" the status of a *sup* in accord with (107).

3.3 Verb Inflection

3.3.1 Vedic Versus Classical Sanskrit

Vedic verbs are inflected for person, number, mood, tense/aspect and voice. Finite verbs distinguish all these categories. Participles distinguish only tense/aspect and voice. Infinitives do not distinguish any of them.

A root can form up to four tense/aspect stems (though not every root has all four of them).

(109) a. The present stem *(pác-a-)*
b. The aorist stem *((a-)pā́k-ṣ-)*
c. The perfect stem *(pá-pac-)*
d. The future stem *(pák-ṣyá-)*

The perfect stem is formed by reduplication; the others are formed by suffixation. Every tense/aspect stem can be directly inflected for person (first, second, third), number (singular, dual, plural), and voice (active, middle) to make a complete finite paradigm, or it can undergo other affixation processes.

The present, aorist, and perfect stems are each inflected with a distinct set of person/number endings; the future stem is inflected exactly like the present stem. Present, aorist, and perfect (but not the future) distinguish four moods:

(110) a. Indicative
b. Optative
c. Subjunctive
d. Imperative

The inflection of the present stem marks a distinction between present tense and "imperfect" tense, which in spite of its name is not imperfective or progressive but simply a preterite. Imperfect tense has no modal or participial forms; its inflection resembles that of the aorist in having a prefixed augment and partly in the form of its person/number endings. Altogether, then, there are five tenses.

Person/number endings and participial endings mark a distinction between active and middle voice throughout. Middle voice has a reflexive function for some verbs, and passive verbs always take middle endings. However, many verbs simply require the middle endings for no particular reason. The present stem, moreover, marks passive by a special stem-forming suffix.

The restriction that the future and imperfect have no modal forms is curious. Why should modality not be distinctive in the future and imperfect, but only in the present, aorist, and perfect? The answer may be that the future and imperfect are pure tenses in the sense that they locate the time of an eventuality after and before speech time, respectively. Aorist and perfect, on the other hand, are *relative tenses* (or *aspects* in the Reichenbachian sense) which locate the time of an eventuality in relation to a *reference* time, which must itself be fixed in relation to speech time. In modal contexts, only the aspectual component of the aorist and perfect surfaces, not the temporal component. Since the future and imperfect don't have one, they have no moods. Suppose that pure tenses and moods turn predicates into propositions, while relative tenses or aspects are predicate modifiers (i.e. they turn predicates into other predicates). Then it would follow that pure tenses cannot be in the scope of modals, whereas relative tenses or aspects can.

This hypothesis is confirmed by a radical change in the tense/aspect system of Pāṇini's Sanskrit. It differs from the Vedic one in two respects. First, the perfect and the aorist are pure tenses, even with a modal component. The imperfect and perfect refer to non-current, historical past *(anadyatane)*, with the perfect furthermore specialized for reports of hearsay events *(paro 'kṣe)*. In reference to recent past events, the aorist is obligatory. Thus, a temporal opposition between near past and remote past, and a category of evidentiality (hearsay vs. witnessed) — more related to mood than to tense or aspect. The Vedic resultative aorist, as well as the aorist of relative anteriority, disappear, and the perfect loses its generic/habitual reading.

Secondly, there are no modal distinctions outside of the present. The perfect subjunctives, optatives, and imperatives of Vedic, as well as its aorist subjunctives, optatives, and imperatives, disappear.

On the assumption that pure tense cannot be modalized, the second change can be seen as a consequence of the first one. The loss of the aspectual function of the perfect and aorist entails the loss of their modal inflection as well.

At this stage, two new moods are introduced. Part of the morphological residue of the former aorist optative is refashioned as a new **precative** (also called **benedictive**) mood. The second new modal form is the **conditional**.

Formally it is a past of the future, made by inflecting the future stem with the imperfect endings. Although it is morphologically related to the future exactly like the imperfect is related to the present, it is functionally neither a future nor a past, but used for counterfactual (or sometimes just hypothetical) conditionals (like English subjunctive use of *would*).

The inventory of tenses was also enriched. Corresponding to the distinction between near past (aorist) and remote past (perfect and imperfect), a distinction arose between near future and remote future. The new remote future tense was created by grammaticalizing periphrastic expressions consisting of an agent noun in *-tar* (plus the inflected copula, in the first and second person).

The result of all these changes is a system which Pāṇini treats in terms of ten mutually exclusive tense/mood categories, each represented by an abstract affix which is spelled out morphologically in accord with agreement and other constraints. The names of the abstract affixes are really bundles of diacritic markers that encode some of their important shared morphological features. For example, the future and the conditional are *lṚṬ* and *lṚṄ*, and the marker *-Ṛ-* that they uniquely share triggers the introduction of stem-marking morpheme *-sya* after them. The fact that conditional mood *(lṚṄ)* also shares morphological properties with optative mood, imperfect tense, and aorist tense (for example, they take the so-called secondary endings) is captured by assigning these the respective names *lIṄ*, *lAṄ*, and *lUṄ* and letting the rules responsible for their shared morphology be triggered by the marker *-Ṅ* that all four categories uniquely share. The fact that future tense *(lṚṬ)* also has unique morphological properties in common with the present, the perfect, the remote future, the subjunctive, and the imperative, is captured by assigning these the respective names *lAṬ*, *lIṬ*, *lUṬ*, *lEṬ*, *loṬ*, and letting the rules responsible for their shared morphology be triggered by the marker *-Ṭ* that all six of them share.

These markers play a role only in the mapping to morphology. They are not suited for handling such functional and semantic affinities as exist between the ten abstract tense/mood affixes. These are captured by grouping the rules that introduce the *l-* affixes under common headings. For example, the three past tense affixes (aorist *lUṄ*, imperfect *lAṄ*, and perfect *lIṬ*) are assigned under the heading (113) 3.2.84 **bhūte** 'in reference to past time', which takes various nominal affixes under its scope as well.[6]

3.3.2 The *l*-Suffixes

There is much overlap and competition of meaning and use among the ten tenses and moods. All the tense/mood suffixes come under the headings (10) 3.1.1 **pratyayaḥ**, (11) 3.1.2 **paraś ca**, 3.1.91 **dhātoḥ**, which define them as *pratyayas* and ensure that they are placed after verbal roots.

---

[6] As predicted by the proposal in the text, the time reference of the latter is always relative. For example, in the sentence *agniṣṭomayājy asya putro bhavitā* 'his son will be someone who has sacrificed the *agniṣṭoma*', the past time reference of the suffix *-in* in *agniṣṭomayājin-* must be understood in relation to the future time reference of *bhavitā* 'will become'.

Present tense *(laṭ)* is introduced by:

(111) 3.2.123 वर्तमाने लट्
vartamāne laṭ
present-Loc *lAṬ*
'to denote ongoing time, *lAṬ* (present tense) is used'

Additional rules are concerned with extended senses of present tense. E.g. (112) records that the near future and the near past tend to be treated as ongoing time.

(112) 3.3.131 वर्तमानसामीप्ये वर्तमानवद्वा
vartamānasāmīpye vartamānavad vā
present-vicinity-Loc present-like optionally
'time near the ongoing is optionally *(vā)* denoted the same way as ongoing time'

In a *vārttika* on (111) 3.2.123, Kātyāyana notes that the rule must be augmented in order to account for the durative present: **pravṛttasyāvirāme śiṣyā bhavanty avartamānatvāt** 'present tense *(bhavantī)* must be specified for actions that have begun but not ended, because they are not (necessarily) going on (at the time of utterance)'. For example, one can truthfully say *ihādhīmahe* 'we are studying here' even though one is not studying while actually uttering that sentence. One must however have begun to study before that time and expect to continue studying after it. This use of present tense is already allowed by (112), so Kātyāyana's point is that it is not merely allowed but obligatory in these cases.

The three past tenses *lUṄ, lAṄ, lIṬ* tenses come under the heading (113):

(113) 3.2.84 भूते
bhūte
past-Loc
'in reference to past time, (the following) are suffixed to the root'

This heading extends not only over the past tenses, but also over a number of *kṛt* suffixes that locate an eventuality in past time, until it is canceled by (111) 3.2.123 **vartamāne laṭ**.

The aorist *(lUṄ)* is introduced by rule (114), as the default past.

(114) 3.2.110 लुङ्
luṅ (3.2.84 bhūte)
lUṄ
'in reference to past time, *lUṄ* (aorist tense) is suffixed to the root'

Imperfect tense *(lAṄ)* is limited to the remoter past.

(115) 3.2.111 अनद्यतने लङ्
anadyatane laṅ (3.2.84 bhūte)
non-current-Loc *lAṄ*
'in reference to non-recent past time, *laṅ* (imperfect tense) is suffixed to the root'

So is the perfect *(lIṬ)*, but with a further condition that it must be a hearsay report.

(116) 3.2.115 परोक्षे लिट्
parokṣe lit̤ (111 anadyatane) (3.2.84 bhūte)
non-witnessed-Loc *lIṬ*
'in reference to non-recent non-witnessed past time, *lIṬ* (perfect tense) is suffixed to the root'

Are rules (114), (115) and (116) in a blocking relationship? This is a thorny question. According to the normal principles of interpretation of Pāṇini's grammar, the competition between three incompatible morphological elements should induce blocking. The perfect should block the imperfect, and the imperfect in turn should block the aorist. On this understanding, the aorist would refer only to eventualities that have happened earlier during the present day, the imperfect only to witnessed eventualities that have happened before the present day, and the perfect to non-witnessed eventualities that have happened before the present day.

Several specific formulations of Pāṇini's rules also presuppose such a blocking relation. If the aorist were not restricted to the recent past, there would be no sense in a special rule such as (117), which specifically allows for the optional use of the aorist for the remote past in a certain context (Subrahmanyam 1999: 282).

(117) 3.2.122 पुरि लुङ्चास्मे
puri luṅ cāsme (121 vibhāṣā) (84 bhūte)
purā-Loc *lUṄ* and not-*sma*-Loc
'in reference to remote past time, in combination with *purā*, in the absence of *sma*, also aorist tense (in addition to present tense or imperfect tense) is optionally (*vibhāṣā*) suffixed to a root'

In post-Pāṇinian usage, the aorist can refer to any past past event, whereas the imperfect and perfect are restricted as described above. Apparently, linguistic change has undone the original blocking relationship.

3.3.3 Person and Number Endings

The generalized tense/mood suffixes are replaced by the specific *tiṄ* endings which mark person and number.

(118) 3.4.77 **lasya**

(119) 3.4.78

Sg.	Du.	Pl.	
tiP	tas	jhi	(Active 3.p., *parasmaipada, prathama*)
siP	thas	tha	(Active 2.p., *parasmaipada, madhyama*)
miP	vas	mas	(Active 1.p., *parasmaipada, uttama*)
ta	ātām	jha	(Mediopassive 3.p., *ātmanepada, prathama*)
thās	āthām	dhvam	(Mediopassive 2.p., *ātmanepada, madhyama*)
iṬ	vahi	mahiṄ	(Mediopassive 1.p., *ātmanepada, uttama*)

These endings are themselves in turn subject to a variety of allomorphic replacements and phonological processes. The basic forms correspond to the

'primary' endings (used in the present tense, among others) in the active *(parasmaipada)*, and to the 'secondary' endings (used in the imperfect tense, among others) in the mediopassive *(ātmanepada)*. This choice of basic forms allows the simplest rules for deriving the allomorphs.

The principle that substitutes are treated like the original ((107) 1.1.56 **sthānivad ādeśo 'nalvidhau**) is very important here. It dictates that *tiN̄* substitutes inherit the properties of the *l*-endings they replace. For example, a *tiP* that replaces *lAṬ* counts as having the marker Ṭ. Similarly, the ending *ṆaL* that replaces *tiP* in the perfect counts as a *tiN̄* ending, even though it is not contained in the list (118).

3.4 Stem Formation: The *vikaraṇas*

3.4.1 The Tense/Mood Stems

Before those *l*-endings which belong to the *sārvadhātuka* class, a stem-forming element *(vikaraṇa)* is inserted, which together with the root constitutes the *an̄ga* of the ending (by (98)). Which particular *vikaraṇa* is inserted depends on three things:

- tense/mood,
- voice (diathesis)
- the verb's conjugational class (in the present active only)

Each *l*-element represents a different tense/mood category. The three voices are determined by whether the endings have been chosen to denote the Agent, the Goal, or the Process (section 2.2). The ten conjugational classes are determined by the listing of verb roots in the *dhātupāṭha*.

The *sārvadhātuka l*-endings fall into two classes, the first of which roughly corresponds to the present system in the terminology of Western linguistics:

(120) a. *Present system: laṭ* 'present', *leṭ* 'subjunctive', *loṭ* 'imperative', *lan̄* 'imperfect', *lin̄* (optative in the *vidhi* 'hortatory/imperative' function)
b. *Non-present systems: lun̄* 'aorist', *luṭ* 'periphrastic (remote) future', *lṛṭ* '(near) future', *lṛn̄* 'conditional'

Each set of *sārvadhātuka l*-endings of the non-present system determines its own *vikaraṇa* in a straightforward way:

(121) a. Before *lun̄* (aorist) endings: *cli*
b. Before *luṭ* (remote future) endings: *tāsI*
c. Before *lṛṭ* (future) and *lṛn̄* (conditional) endings: *sya*

The aorist *vikaraṇa cli* is an abstract placeholder. It is always replaced by one of the actual endings according to the phonological shape and lexical identity of the root.

The *vikaraṇas* bear grammatical markers which among other things have effects on the form of the root, particularly on its accent and strong vs. weak grade, e.g. *kṛṣ+ŚaP+tiP* → *kárṣati* (first class), but *kṛṣ+Śa+tiP* → *kṛṣáti* (sixth class).

Also, they are themselves subject to alternations of stress and strong vs. weak grade, e.g. *su-nu-tiP* → *sunóti*, but *su-nu-mas* → *sunumás*.

The table on the next page shows the system of tense/mood stems in outline.

Before *l*-endings belonging to the present system, the choice of *vikaraṇa* depends on voice, and in the active, on the verb's conjugational class. If the *l*-endings denote the Goal or the Process, *yaK* is inserted by [122].

(122) 3.1.67 सार्वधातुके यक्
sārvadhātuke yak (66 bhāvakarmaṇoḥ)
sārvadhātuka-Loc *yaK*-Nom
'*yaK* is added before a *sārvadhātuka* suffix which denotes the Goal or the Process'.

The marker *K* on *yaK* blocks *guṇa* on the root by 7.3.84 *sārvadhātukārdhadhātukayoḥ*, in virtue of the prohibition 1.1.5 *kṅiti ca*. Thus we get the passive form *bhū-yá-te*, not **bhav-yá-te* etc.

If the *l*-endings denote the Agent, one of a set of other *vikaraṇas* is inserted before them. The choice depends on the conjugation class of the verb, which is an unpredictable lexical matter. The roots are listed in the *dhātupāṭha* in ten groups, each of which constitutes a different class. The default *vikaraṇa* is *ŚaP*, which is inserted by [123].

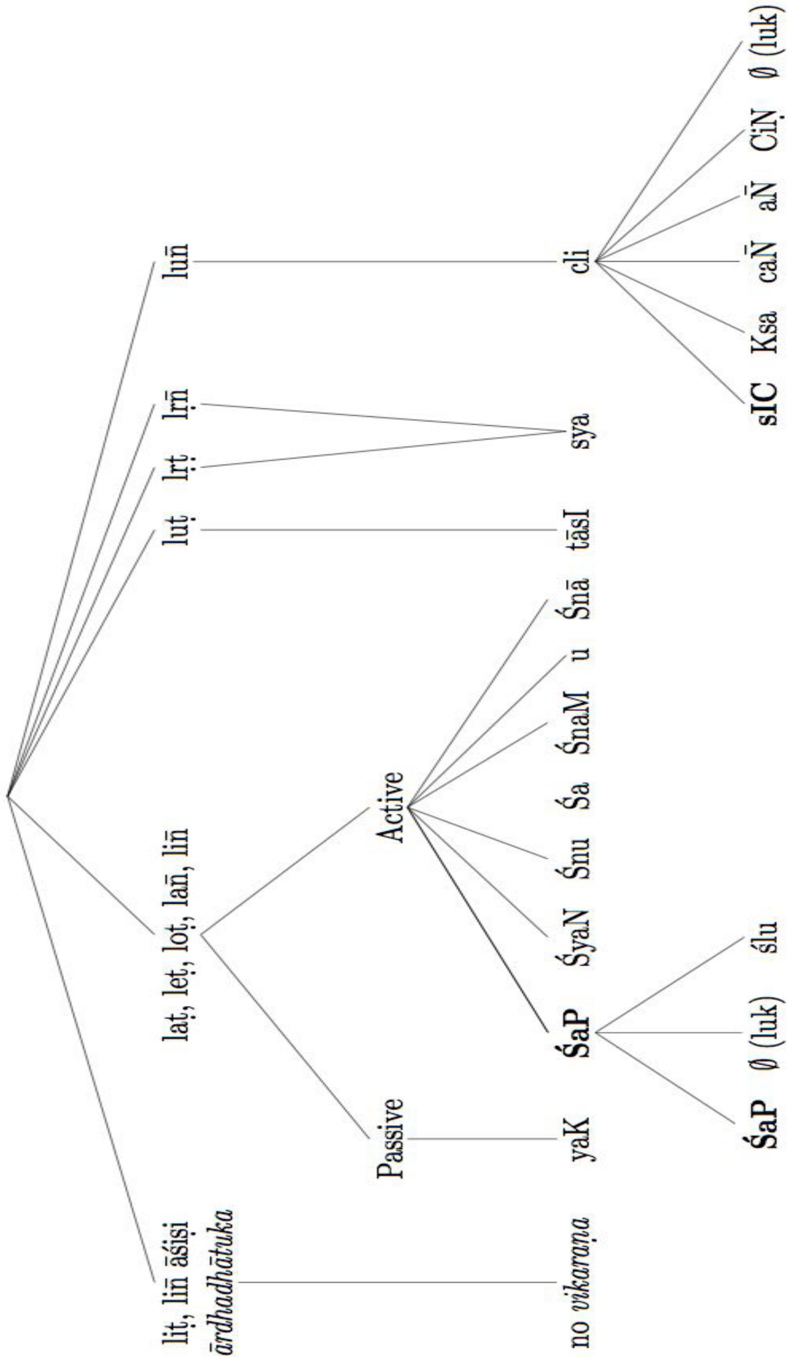
(123) 3.1.68 कर्तरि शप्
kartari śap (67 sārvadhātuke)
Agent-Loc *ŚaP*-Nom
'*ŚaP* is suffixed to the root before a *sārvadhātuka* suffix which denotes the Agent.'

ŚaP is added to roots of several conjugational classes: the first (the *bhū* class), the second (the *ad* class), the third (the *hu* class), the tenth (the *cur* class). It is also added to derived roots, including causatives and others with the suffix *ṆiC*, desideratives (*saN*), intensives (*yaṆ*), and denominal verbs, which all get the designation *dhātu* 'root' by (100) 3.1.32 **sanādyantā dhātavaḥ**.

In roots of the second and third class, *ŚaP* is deleted, or, more precisely, replaced by one of the two null elements *luK* and *Ślu*. They do not inherit the properties of *ŚaP*, because (124) 1.1.63 **na lumatāṅgasya** stipulates that deletion effected through replacement by these elements is not subject to (104), in effect reinstating the *siddha*-principle.

(124) 1.1.63 न लुमताङ्गस्य
na lumatāṅgasya (1.1.62 pratyayalope pratyayalakṣaṇam)
not *lu*-containing-Instr *stem-Gen*
'When a suffix is deleted by an element containing *lu*, the operations it triggers on an *aṅga* (stem) do not apply.'

For example, in *i-∅-más* 'we go', even though ∅ *(luK)* is a replacement of *ŚaP*, which triggers *guṇa* by 7.3.84 **sārvadhātukārdhahātukayoḥ**, it does not itself trigger *guṇa*.



liṭ, lin̄ āśiṣi
ārdhadhātuka
laṭ, leṭ, loṭ, lan̄, lin̄
luṭ
lṛṭ
lṛn̄
lun̄
no vikaraṇa
Passive
Active
yaK
ŚaP
ŚyaN
Śnu
Śa
ŚnaM
u
Śnā
tāsI
sya
cli
ŚaP
∅ (luk)
ślu
sIC
Ksa
caN̄
aN̄
CiṆ
∅ (luk)

Other classes of verbs receive different *vikaraṇas* by a set of special rules (*apavādas*) which block [123].

The marker *P* on *Śa P* has several functions. One of its functions is to activate rule [125], which blocks it from getting accented by 3.1.3 *ādyudāttaś ca* (rule (12)).

(125) अनुदात्तौ सुप्पितौ
anudāttau suppitau
unaccented-DuNom *suP-Pit*-DuNom
'*Sup* endings (case-number endings) and endings marked with *P* are unaccented'.

Accent is then assigned by:

(126) 6.1.162 धातोः
dhātoḥ (159 antodāttaḥ)
root-Gen
'The final syllable of a root bears an *udātta* pitch accent.'

Another function of the marker *P* in *ŚaP* is to preclude the assignment of the marker *Ṅ* to it by [127].

(127) 1.2.4 सार्वधातुकमपित्
sārvadhātukam apit (1 ṅit)
sārvadhātuka-Nom non*P-it*-Nom
'a *sārvadhātuka* which does not have the marker *P* has the marker *Ṅ*'

If this redundancy rule were to apply, it would result in an undesired prohibition of *guṇa* by 1.1.5 *kṅiti ca*. The marker *P* blocks it, thus ensuring that forms like *bhū-ŚaP-tiP* → *bháv-a-ti* are correctly derived.

The marker *Ś* ensures that *ŚaP* is classed as *sārvadhātuka* (3.4.113 **tiṅśit sārvadhātukam**). Another function of the marker *Ś* on *ŚaP* is to block rule [128] from replacing *ai* by *ā* before it in cases like *gai-ŚaP-tiP* → *gāyati* (↛ **gāti*).

(128) 6.1.45 आदेच उपदेशे ऽशिति
ād eca upadeśe 'śiti (6.1.8 dhātoḥ)
āT-Nom *eC*-Gen basic-form-Loc non-*Ś-it*-Loc
'An underlying root-final *eC* (a diphthong) is replaced by *ā*, except before an item with the marker *Ś*'.

The abovementioned *vikaraṇas* are inserted before *sārvadhātuka* suffixes representing the present system, listed in (120a). The remaining tense/mood categories are formed off distinct stems. Let's take a brief look just at the aorist verb form *apāci* in (2).

Before the *l*-endings representing the aorist (*lUṄ*), instead of *ŚaP* and the other *vikaraṇas*, the element *CLI* is inserted before them (independently of whether they denote Goal, Process, or Agent) by [129]. In other words, voice is neutralized in the aorist stem. This is done by rule (129), which blocks (123) 3.1.68 **kartari ŚaP**.

(129) 3.1.43 च्लि लुङि
cli luṅi
Cli-Nom *lUṄ*-Loc
'before (the endings replacing) *lUṄ*, *Cli* is inserted'.

Cli is in turn replaced by the specific aorist markers *sIC*, *Ksa*, *CaṄ*, *aṄ*, *CiṆ*, or deleted, under partly phonological, partly lexical conditions.

The verb *apāci* in (2) is special aorist passive form, which is restricted to the third person singular, and triggers deletion of the person/number ending after it:

(130) 3.1.66 चिण् भावकर्मणोः
ciṇ bhāvakarmaṇoḥ (44 cleḥ)
CiṆ-Nom Process-Goal-DuLoc
'To express the Process or the Goal (i.e. in the passive and in the impersonal passive), *CLI* is replaced by *CiṆ*.'

(131) 6.4.104 चिणो लुक्
ciṇo luk
Ciṇ-Gen *luk*
'After the aorist suffix *Ciṇ*, person/number endings are deleted.'

E.g. *pac-CLI* → *a-pāc-i-ta* → *a-pāc-i* 'was cooked'.

3.5 Morphological Lessons of Pāṇini's Grammar

3.5.1 Blocking and Substitution

As you will have noticed, the distribution of suffixes and the alternations in their shapes are bewilderingly complex; yet Pāṇini succeeds in extracting some fairly general patterns.

In addition to these regularities, some roots and affixes are subject to idiosyncratic alternations in various morphological contexts. There are two ways to handle such allomorphy in the grammar: blocking and replacement. In either case, one form in a set of alternating forms is chosen as basic, in such a way as to allow the simplest overall description. In the substitution method, the basic form is introduced by a general rule everywhere and then replaced by the other alternants in specific contexts. In the blocking method, the basic form is introduced by a general rule and the alternants by special rules which block the general rule in specific contexts. There is a close conceptual relationship between these two procedures, of which the tradition is well aware.

Normally the simplest description results if an actually occurring form is chosen as underlying form, and among the actually occurring forms the one with the widest distribution. But sometimes there are reasons to prefer a form with more restricted distribution, or even to posit an underlying form which does not occur as an actual form at all.

Pāṇini typically (though by no means exclusively) uses blocking in derivational morphology, and substitution in inflectional morphology. The main reason for this is that replacements by convention inherit the morphological properties

of the elements they replace (for example, they have the same effects on the vowel shape and accent of the stem to which they are added). In Sanskrit, at least, these properties are typically invariant in inflectional alternants, but vary in derivational alternants (presumably at least in part because paradigmatic leveling is more frequent within an inflectional paradigm).

The replacement and blocking techniques can also be combined. This method involves setting up a wholly abstract underlying form, and a rule replacing it by the basic allomorph, which in turn is blocked by the special allomorph. An example of this is Pāṇini's treatment of the aorist formative, which has several variants: *s, (sIC)*, (the basic allomorph), *sa*, *a* (with or without reduplication), and zero. These are all derived from an underlying *cli* which never surfaces in that form. It is always replaced, either by the basic allomorph *sIC* through a context-free rule, or in specific contexts by other allomorphs through rules like (130) which block *sIC*. The zero ending however is derived from *sIC* itself by replacement rules which substitute the null element *luk* for it.

3.5.2 The Nature of Sanskrit Allomorphy

Pāṇini's grammar reveals two important morphological generalization about Sanskrit. First, *the locus of suppletion is the morpheme:* all suppletion in Sanskrit verb inflection is either root suppletion or suffix suppletion. There is no "multi-morpheme suppletion" whereby sequences of morphemes are idiosyncratically replaced by other sequences of morphemes, and there is no "total suppletion" of entire words.

Secondly, the distribution of suppletive allomorphs is determined by the same contextual factors that determine the distribution of morphemes themselves, that is, by prosodic phonological conditions (from a contemporary perspective, syllable structure and accent) and by morphological classes.

A paradigm-centered approach could not capture either of these absolutely central generalizations.

3.5.3 Multiple Exponence and Null Exponence

Stump (2001) draws a distinction between what he calls realizational theories and incremental theories of morphology. Realizational theories hold that words are built up by spelling out features as affixes, while incremental theories hold that words are built up by percolating the features of affixes to the stem+affix combinations they enter into. According to Stump, the basic empirical issue that divides the theories is that incremental theories privilege one-to-one correspondences between morphemes and morphosyntactic features or feature bundles. Each morpheme of a word would be expected to correspond to a subset of its morphosyntactic features, and cases where several morphemes correspond to a single feature bundle, and cases where no morpheme corresponds to a feature bundle, would represent descriptive complications. On the realizational view, there is no such expectation. Where several morphemes correspond to a single feature bundle, there are simply several realization rules, and where no morpheme corresponds to a feature bundle, there is simply no realization rule. On the face of it, for example, for example, in the 3.Sg. aorist form *á-kār-ṣ-am*, one

might see four realizations of the aorist category: the augment *a-*, the lengthening of the root vowel, the suffix *-s-*, and the *-am* allomorph of the 1.Sg. ending.

Stump argues in favor of the realizational view on the grounds that the incremental view imposes arbitrary choices in cases of multiple exponence, and artificial solutions in cases of null exponence. Pāṇini's grammar of Sanskrit tends to show the opposite. Technically, it may be considered an incremental theory. Yet it adopts the one-to-one correspondence between morphological elements and morphosyntactic features as the baseline, entirely for reasons of descriptive simplicity. In nearly all cases of apparent multiple exponence, one of the morphemes turns out to be the bearer of the morphosyntactic feature bundle, and the others have a different function. Consider again *á-kār-ṣ-am*. The augment *a-* is added to three tense/mood categories (imperfect, aorist, and conditional), and (as Vedic shows) only in finite inflection. Thus, it is not a marker of the aorist. The lengthening of the root is a morphophonological process which is triggered not only in the aorist, but in a vast class of morphological categories under certain phonological conditions. The person/number allomorph *-am* is shared with the imperfect, the optative, and the precative, and the conditional, and also marks active voice. The "real" marker of the aorist in this class of aorists, then, is *-s-*. A similar argument can be given in almost every case of "multiple exponence" in Sanskrit. At least in this analysis, there is no arbitrariness and, at the theoretical level, no multiple exponence.

As for null exponence, Pāṇini's grammar reveals the exceptionless generalization that *null morphemes are always allomorphs of overt morphemes.* His morphological empty elements (*luk* etc.) are needed only as replacements of suffixes with phonological substance, never as morphemes in their own right. On the realizational view, there is no reason why that should be so.

I conclude that Pāṇini's descriptive practice constitutes evidence against the realizational view and in support of the incremental view, in so far as the one-to-one correspondence between morphemes and morphosyntactic feature bundles is a natural consequence of the latter but not of the former.

3.6 Derivational Morphology

3.6.1 The *Taddhita* Section

The treatment of secondary nominal derivation occupies almost a quarter of the *Aṣṭādhyāyī* and has an intricate structure. As mentioned, one of its most interesting features is that Pāṇini's technique enables the rules of suffixation to be separated from the rules of meaning assignment. Ingeniously exploiting this device in the *taddhita* section to deal with the competition among multiply polysemous suffixes, Pāṇini organizes the section as follows.

(132) Suffix$_1$

Suffixes which block Suffix$_1$ in all of its meanings:

Suffix$_{1'}$ with stem classes X$_{1'}$, Y$_{1'}$, ...

Suffix$_{1''}$ with stem classes X$_{1''}$, Y$_{1''}$, ...

etc.

Meaning$_{1a}$ of Suffix$_1$, Suffix$_{1'}$, Suffix$_{1''}$, ...

Suffixes which block Suffix$_1$ in Meaning$_{1a}$:
Suffix$_{1a'}$ with stem classes $X_{1a'}$, $Y_{1a'}$, ...
Suffix$_{1a''}$ with stems $X_{1a''}$, $Y_{1a''}$, ...
etc.

Meaning$_{1b}$ of Suffix$_1$, Suffix$_{1'}$, Suffix$_{1''}$, ...

Suffixes which block Suffix$_1$ in Meaning$_{1b}$:
Suffix$_{1b'}$ with stem classes $X_{1b'}$, $Y_{1b'}$, ...
etc.

(Repeat for Suffix$_2$, Suffix$_3$, ...)

3.6.2 Compounding

All compounds are derived by combining *padas*, each of which must have its own case ending, which is then deleted by [133]:

(133) 2.4.71 सुपो धातुप्रातिपदिकयोः
supo dhātuprātipadikayoḥ (58 luk)
suP-Gen root-stem-DuGen
'case endings in roots and stems are deleted'

The reason for this analysis is that it simplifies the morphological derivation of compounds. First, in some types of compounds the case endings are actually retained; these can simply be characterized as exceptions to deletion. Secondly, it accounts for the word status of each constituent by rule [134] (= [103]).

(134) 1.4.14 सुप्तिङन्तं पदम्
sup-tiṅ-antam padam
suP-*tiṄ*-ending-Nom word-Nom
'An element tha ends in *suP* or *tiṄ* is (termed) *pada* 'word'.

The word status of each member of a compound is required by the phonology. For example, in *rājapuruṣa* 'king's servant' the first member *rājan*-, being a word, loses its final -*n* by rule [41].

Compounding rules are of the form

(135) A_{Nom} B_{Instr} = 'A is compounded with B'

where the nominative item is called *upasarjana* and is positioned first in the resulting compound. For example, rule [136] compounds a genitive with its head:[7]

[7] In this rule, the tradition wrongly continues *vibhāṣā* from 2.1.11; for discussion see Kiparsky 1979, 39, Joshi and Bhate 1984, 95.

(136) 2.2.8 षष्ठी
ṣaṣṭhī **(2.1.2-4 sup, samāsaḥ, saha supā) (2.1.18 vā) (2.1.22**
sixth-Nom
tatpuruṣaḥ)

'[an item ending in] a genitive case suffix is (preferably) [compounded] (with [an item ending in] a case suffix) (to form a *tatpuruṣa*)'.

yielding such nominal stems as [[aśva+sya][pati+Su]] "horse-lord", which then by [133] lose their internal case endings, and get inflected with external case endings like any other nominal stem.

Compounds fall under the constraint (137) which governs all word formation.

(137) 2.1.1 समर्थः पदविधिः
samarthaḥ **padavidhiḥ**
semantically-related-Nom word-operation-Nom
"An operation on words is semantically related.'

The wording is not quite clear, but the rule is evidently designed to insure that compounding processes can combine a word with its modifier or complement, but not with a complement or modifier of another. As Patañjali points out, an external modifier cannot be ordinarily be construed with a member of a compound. For example, the compound *rājapuruṣaḥ* is analyzed as *rājñaḥ puruṣaḥ* 'king's servant', but the expression *ṛddhasya rājapuruṣaḥ* does not mean 'servant of a rich king', i.e. it cannot have the semantic bracketing ((ṛddhasya rājñaḥ) puruṣaḥ).

Patañjali on 2.1.1 discusses a number of interesting cases where compounds apparently violate this word integrity principle (Joshi & Roodbergen p. 35 ff.). They have not been systematically studied in modern grammar as far as I know. The examples seem to fall into certain natural classes.

One group consists of cases where an external modifier is construed with a governed member of a compound that bears an intrinsic relation to the governing member.

(138) **Devadattasya gurukulam** **(= kulaṃ guror Devadattasya)**
Devadatta-Gen teacher-family-Nom
'the family of Devadatta's teacher' *(lit.)* 'Devadatta's teacher-family'

These apparent syntax/morphology mismatches should probably be treated at the level of semantics. A semantic inheritance mechanism whereby properties of individuals become properties of groups to which those individuals belong is needed in any case. For example, *a laughing group of children* is really a group of laughing children: it is not the group that laughs, but the individual children that it consists of. Similarly, in (138) the property of being Devadatta's has been inherited from the teacher by the teacher's family. The cases where external modifiers are precluded are those where, on semantic grounds, such inheritance makes no sense. For example, the property of being rich is not inherited from a king by his servants.

In another group of cases, commonly found in literature, the external modifier seems to modify part of a compound whose head is a numeral or measure:

(139) **saktvāḍhakam āpāṇīyānām**
barley-measure-Nom for-sale-GenPl
'a measure of barley grains for sale' (Patañjali)

This is presumably to be analyzed as 'a barley-grain measure of a thousand (barley-grains)', like English *a student population of 1000.*

A third group mentioned by Patañjali seems to have a somewhat different character. It involves compounding of the negation prefix *a*-.

(140) **amāṣaṃ haramāṇam**
non-lentil-Acc taking-Nom
'not taking (even) a lentil' (Patañjali)

Similarly, *a-śabdam kurvan* 'not making (so much as) a sound' (*Āp.ŚS* 6.11.4) In these examples, literally 'taking a non-lentil', 'making a non-sound', the negation must clearly be construed with the verb, but is expressed on its object. This kind of negation can be construed as NP-negation forming an expression denoting a minimal amount, which is then interpreted like a negative polarity item, viz. 'taking not-(even-)a-lentil' = 'taking very little'. Again, once the semantics is understood, there is no need to assume a morphology/syntax mismatch.

Interestingly, Wh-pronouns in Sanskrit can be freely compounded (like any other pronoun).

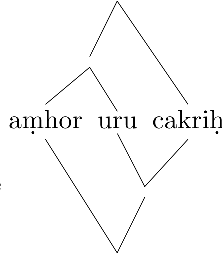
(141) **kiṃgotro nv aham asmīti? sāham etan na veda**
'what(Q)-family Part I am-Quote? she I this not know-1Sg
yadgotras tvam asi
what(Rel)-family you are.'
'So, what family am I from?' 'I don't know what family you are from.' (*ChUp* 4.4.1-2)

Words in Sanskrit are not "anaphoric islands", and anaphoric binding in Sanskrit is not a relation between maximal projections. This is just what we would expect since (as discussed above) the antecedent of reflexive pronouns in Sanskrit is normally determined at the level of thematic structure (the highest Theta-role).

Another type of apparent morphology/syntax mismatch has recently been discussed by Stump (2001:14), who says: "a compound's morphological structure needn't be isomorphic to its syntactic structure; in the Sanskrit expression [below], for example, the NP *aṃhór* 'distress (abl sg)' is syntactically dependent on *urú*- 'distance, relief' but is not itself part of the compound *uru-cákriḥ* 'causing relief (nom sg)'." His analysis posits the mismatch in (142).

(142)

Syntactic structure:



Morphological structure

But the fact that the complex nominal predicate *uru-cákriḥ* 'freeing, causing relief' assigns ablative case to its complement like *uru* 'free' does is no more exotic than the fact that the verbs *to relieve* and *to free* take the same kinds of PP complements that *relief* and *free* do (namely 'from' or 'of'). The generalization that such complements are inherited through morphological derivation can be readily dealt with in the morphology itself, and many theories of morphology have spelled out mechanisms for doing so.

In late classical literature, there occur however cases where the first, most deeply embedded member of *bahuvrīhi* compounds have external complements that cannot be explained away semantically.

(143) **pīnābhyāṃ madbhujābhyāṃ**
brawny-Instr my-arms-Instr
bhramitagurugadāghātasaṃcūrṇitoroḥ
whirled-heavy-club-crushed-thigh-Gen
whose thighs have been crushed by the strokes of the heavy club whirled around by my brawny arms (*Harṣacarita* 5.35)

4 Phonology

4.1 Phonological Domains

When there is no specific statement to the contrary, a phonological rule applies if and only if the conditioning context (the "trigger") and the undergoing element (the "target") are adjacent. If some class of elements may intervene between them, this must be specifically stated. Phonological rules may place other conditions on the relation between trigger and target, such as requiring them to form a base+suffix combination, to be within the same word or in the same metrical unit, to be in close contact, or to be semantically related. Each such relation determines a different kind of phonological domain. Rules which have one of these domains in common are listed together, in so far as possible, under a heading that specifies that domain for all of them.

Independently of this topical organization according to domains of application, rules fall into major groups on the basis of constraints on their mutual interaction. Conceptually, these constraints are all restrictions on the *siddha*-principle.

The schema in (1) makes no distinction between 'phonology', 'morphophonology', and 'allomorphy'. And indeed there seems to be no principled distinction made between them in the organization of the grammar. But now that we have seen how many of the theoretical distinctions made in contemporary linguistics emerge as if on their own in the *Aṣṭādhyāyī* because the economy principle forces certain groupings that correspond to them, we should ask whether something like these subdivisions can be discerned in the mapping from level 3 to level 4. In fact, it turns out that several metarules of the grammar stipulate different properties for two kinds of rules, which roughly correspond to allomorphy rules and phonological rules, and that the latter in turn fall into two types which correspond to what we would consider morphophonological and phonological.

The rules in the block from 6.4 through the end of 7 are restricted to combinations of an *aṅga* 'base' plus a suffix, in virtue of the heading (144).

(144) 6.4.1 अङ्गस्य
aṅgasya
base-Gen
'in place of (the final segment of) a base'

The need to restrict some rules in this way is illustrated by the contrast in (145), which shows that vowel sequences can be treated differently at the stem-suffix juncture in (145a) and across a word boundary in (145b).

(145) a. *śrī+as* → *śriyas* 'riches'
b. *śrī+artha+am* → *śryartham* 'for the sake of prosperity'

The -*iy*- in *śriyas* replaces -*ī*- by rule 6.4.77 (146).

(146) 6.4.77 अचि श्नुधातुभ्रुवां य्वोरियङुवङौ
aci śnu-dhātu-bhruvāṃ y-vor iyaṅ-uvaṅau (6.4.1
aC-Loc *śnu*-Root-*bhrū*-PlGen *y*-*v*-DuGen *iy*-*uv*-DuNom
aṅgasya)

'before a vowel, base-(final) *i, u* of (the conjugation marker) *Śnu*, of a root, and of *bhrū* 'brow' are replaced by *iy* and *uv*, respectively'

As a rule of the *aṅga* section, (146) is not applicable across word boundaries, as in (145b).

From our point of view, rule (146) would probably be considered "morphophonological". But many rules of the *aṅga* section also deal with "allomorphy". Recall rule (107), which states an important distinction between two sorts of replacement processes which corresponds to that distinction: non-phonological properties of the input are transferred from inputs to their replacements, while satisfaction of phonological properties is checked on the actual output string only (Joshi and Roodbergen 1985). Properties which depend on markers, therefore, are always carried over. For example, rule (147) replaces the absolutive suffix *Ktvā* by *LyaP* after a compound (prefixed) verb, except for the negation prefix *a(n)*-.

(147) 7.1.37 समासे ऽनञ्पूर्वे क्त्वो ल्यप्
samāse 'nañpūrve ktvo lyap
compound-Loc non-*aÑ*-initial-Loc *Ktvā*-Gen *LyaP*
'in a compound that does not begin with *aÑ*-, *Ktvā* is replaced by *LyaP*'

The suffix *Ktvā* is defined as a *kṛt* suffix (3.4.21). The replacement *LyaP* is introduced in (147), outside of the *kṛt* section, so it not a *kṛt* suffix. But (107) transfers the property of being a *kṛt* suffix from *Ktvā* to its replacement *LyaP*. The desired effect is that rule (148), which introduces the augment *tUK* after a short root before a *kṛt* suffix with the marker *P*, will also apply before *LyaP*.

(148) 6.1.71 ह्रस्वस्य पिति कृति तुक्
hrasvasya piti kṛti tuk
short-Gen *P*-*it*-Loc *kṛt*-Loc *tUK*
'*t* is inserted after a short vowel before a *kṛt* suffix marked with *P*'

E.g. *pra-kṛ-Ktvā* → *pra-kṛ-LyaP* → *pra-kṛt-ya.*

On the other hand, properties such as "having one vowel", or "beginning with a consonant" are not transferred. For example, rule 7.1.84 **diva aut** states that the final consonant of *div* 'the sky' is replaced by *au* before the nominative singular ending *-sU*: *div-s* → *diau-s* (→ 6.1.77 (25) *dyaus*) (compare e.g. dative singular *div-e*). If the output stem preserved the input's property of "ending in a consonant", rule 6.1.68 **halṅyābbhyo dīrghāt sutisyapṛktaṃ hal** would wrongly apply to delete the ending *-s (*dyau).* In other words, the phonological change of *-v* to *-au* must be taken into account when assessing whether the phonological conditions of rule 6.1.68 are satisfied.

Another distinction reminiscent of the morphophonology/allomorphy divide emerges, again on purely technical grounds, between substitutes consisting of one sound and substitutes consisting of more than one sound. The normal form of a phonological rule is:

(149) $A \rightarrow B$ / C__D

where A and B are single segments. The single-segment property of the change, and the adjacency of the triggering context (locality), typically hold not only for purely phonologically conditioned rules, but equally for morphologically conditioned and for lexically conditioned phonological rules, such as the change of *div* to *diau* just mentioned. In the default case, therefore, a substitute consisting of one sound will replace just one segment of the input, and this will be the last sound of the input if the trigger (the conditioning environment) is on the right, and the first sound of the input if the trigger is on the left. For example, *v* → *au* in *div+s* is triggered by the ending, so it is the *last* sound of the stem which is affected. This generalization is exploited to simplify the formulation of grammatical rules, by supplying the default behavior as a convention:

(150) 1.1.52 अलो ऽन्त्यस्य
alo 'ntyasya (50 sthāne) (49 ṣaṣṭhī)
aL-Gen last-Gen

'(a substitute replaces) the last sound (of a substituend specified in the genitive)'

(151) 1.1.54 आदेः परस्य
ādeḥ parasya (52 alaḥ) (50 sthāne)
initial-Gen following-Gen
'(a substitute replaces) the first (segment) (of a substituend) which follows'

A substitute consisting of more than one sound, on the other hand, typically replaces the whole input (because such a substitution most likely is an allomorphy process, we would say). For example, (147) **samāse 'nañpūrve ktvo lyap** substitutes *LyaP* for the *entire* ending *Ktvā*.

Naturally, substitutes that go against the default generalizations must be marked. Such contrary cases occur in both directions, and each is flagged by its own marker. "Short" substitutes that (contrary to the default) replace the entire input are marked by *Ś*, and "long" substitutes that (contrary to the default) replace just one sound of the input are marked by *N̄*.

To repeat, there is no reason to believe that Pāṇini had any principled rule typology analogous to those developed in many modern linguistic theories. He simply dealt with the morphology/phonology interface phenomena of Sanskrit by means of his usual grammatical technique, driven solely by the simplifying, generalizing imperative. Yet by consistently applying this technique he ended up framing conventions such as (107) and (150)-(151), which in their own way reflect approximately the distinction between what we would call allomorphic and morphophonological rules, on the basis of their purely formal properties.

Within phonological rules, other major classes emerge. Those which are restricted to apply in close contact *(saṃhitā)* are termed *sandhi* rules (from *samdhā-* 'put together, join'). The most important *sandhi* rules are in three groups, each headed by *saṃhitāyām* 'in close contact' (6.1.72–157, 6.3.114–139, 8.2.108–8.4.68). Smaller blocks of rules are limited to applying anywhere within the domain of a *pada* 'word' (8.4.1 ff.), and to the domain of a metrical *pāda* 'verse' (8.3.9 ff.). Words and metrical verses are also exactly the domains whose *edges* can block or condition the application of phonological rules. The absence or presence of a word on the left or right defines a sentence-initial or non-sentence-initial environment. Intonation rules can be semantically conditioned by a trigger which need not be in close contact to the undergoer, or even adjacent to it. A few rules require *both* close contact *and* semantic relationship between trigger and undergoer. The joint requirement of close contact and semantic relationship defines a domain which can be identified as the phonological phrase. Within this domain, certain sandhi processes that otherwise apply in close contact are suspended.

4.2 Types of Rule Interaction

The second major criterion by which rules are grouped is by shared constraints on their mutual interaction. Just as string adjacency is the unstated default relation between trigger and target, and those cases where some other relation between them obtains are specially provided for in the grammar, so there are

unstated default principles governing rule interaction, and those interactions which diverge from the default are specially provided for. What precisely the unstated default principles governing rule interaction are, however, is a matter of some controversy. As stated in section 1, my view is that Pāṇini assumed the *siddha*-principle, the word-integrity principle, and the blocking principle. Before proceeding I summarize what the tradition says on this matter.

4.2.1 The Traditional View

Traditionally the order of application of rules in the *Aṣṭādhyāyī* is determined by a hierarchy of four principles:

(152) Rule A supersedes rule B under the following conditions:
 a. if A follows B in the *Aṣṭādhyāyī* (A is *para)*,
 b. if A is applicable whether or not B applies (A is *nitya)*,
 c. if A is conditioned internally to B (A is *antaraṅga)*,
 d. if the inputs to which A is applicable are a proper subset of the inputs to which B is applicable (A is *apavāda)*.

The principles in (152) are assumed to form a hierarchy of increasing strength, so that [152b] takes precedence over [152a], [152c] over [152b], etc. Except for (152a) (see below), neither the principles nor the hierarchy are stated in the grammar, but several versions of them are made explicit by the traditional commentators. Not included in this list, but tacitly assumed by the tradition, is the fundamental principle that when a rule can apply to the output of another, it does, unless this is blocked by some other principle or rule.

The following paragraphs briefly present the motivation for each principle and for their hierarchy.

The *para* principle (152a) is stated in the grammar in connection with a set of definitional rules which must apply disjunctively. These definitional rules are gathered under the headings [90]-[91]. As was discussed above, this so-called *ekasaṃjñā*-section includes, among many others, the rules that map semantics to thematic roles (*kārakas*), which are placed in that section in order to prevent expressions from being assigned more than one role.

According to tradition, however, the precedence of *para* rules stipulated in (91) holds throughout the grammar.

A is a *nitya* 'constant' rule with respect to B if A is applicable whether or not B applies, but not conversely. A *nitya* rule has precedence over a non-*nitya* rule. This is equivalent to saying that bleeding order has priority over non-bleeding order, so we can call it the bleeding principle.

A simple example of the *nitya* principle is the derivation of *rud-hi* → *rudihi* 'weep!' (2Sg). Underlying *rud-hi* is potentially subject to two rules:

(153) 6.4.101 हुझल्भ्यो हेर्धिः
 hujhalbhyo her dhiḥ (101 aṅgasya)
 hu-jhal-PlAbl *hi*-Gen *dhi*-Nom
 'after (the root) *hu* and after a base ending in an obstruent, -*hi* is replaced by -*dhi*'

(154) 7.2.76 रुदादिभ्यः सार्वधातुके
rudādibhyaḥ sārvadhātuke (35 iṭ valādeḥ)
rud-beginning-PlAbl *sārvadhātuka*-Loc
'after the roots *rud* etc., the augment *iṬ* is inserted before *sārvadhātuka* endings beginning with a *vaL* consonant'

Rule (154) bleeds (153), therefore precedes it.

When the *nitya* principle does *not* give the right results, special countermeasures are taken. Consider again the rules (147) and (148). At the stage *adhi-i-ya* (after *Ktvā* has been replaced by *LyaP* by (147) 7.1.37 **samāse 'nañpūrve ktvo lyap**), two rules are potentially applicable: contraction of a pair of like vowels into a single long vowel by (27) 6.1.101 **akaḥ savarṇe dīrghaḥ**, and insertion of the augment *-t (tUK)* at the end a short-vowel root by (148) 6.1.71 **hrasvasya piti kṛti tuk**. Since the augmentation is conditioned by a short root vowel, it is bled by contraction, which should therefore take effect first, and *t*-augmentation would then be blocked. The resulting form, **adhīya*, is however incorrect. So here the default principle leads to the wrong result. Therefore, a special rule is required which stipulates that vowel contraction (among other processes) is *asiddha* with respect to insertion of the augment *tUK*.

(155) 6.1.86 षत्वतुकोरसिद्धः
ṣatvatukor asiddhaḥ
ṣ-quality-*tUK*-DuLoc not-effected
'(these rules) are treated as not effected with respect to retroflexion of *s* and insertion of the augment *t*-'

The tradition operates with the correlative concepts *bahirañga* 'externally conditioned' versus *antarañga* 'internally conditioned', and posits the principle that *bahirañga* processes are *asiddha* with respect to *antarañga* processes. The tradition knows also a weaker version:

(156) a. **The strong AP:** A *bahirañga* rule is *asiddha* with respect to an *antarañga* rule *(asiddham bahirañgam antarañge)*.
b. **The weak AP:** An *antarañga* rule takes precedence over a *bahirañga* rule *(antarañgaṃ balīyo bhavati)*.

The *antarañga-paribhāṣā* (AP) is reminiscent of cyclic application in generative phonology. Its two versions correspond to two versions of cyclicity, with or without the "Strict Cycle Condition".

The word-integrity principle is a special case of the *antaranga*-principle, but tradition applies the *antaranga*-principle also within words.

4.2.2 A Non-traditional Interpretation

Of the principles in [152], the *para* principle is today generally agreed to be restricted to the *saṃjñā* section (1.4–2.3). Joshi and I (Kiparsky and Joshi 1979, Kiparsky 1982, Joshi in press) have argued at length that the *nitya* principle is subsumed, with the (unstated) master principle that rules apply at any opportunity, under the *siddha*-principle

discussed above in (38). We also argued that Pāṇini did not assume the *antarañga*-principle word-internally, only the word-integrity principle. That is, phrasal rule applications are *asiddha* with respect to rule applications inside words, but rule applications to larger constituents of words are not *asiddha* with respect to rule applications to smaller constituents of words. We showed that the wording of his rules invariably presupposes that the word-internal application of rules is governed by the *siddha*-principle and not by the *antarañga*-principle.

To this argument I would now like to add two new points. The first new point is that Pāṇini *should* have adopted a form of the *antarañga*-principle, i.e. word-internal cyclicity, for it is in fact rather well motivated by phonology/morphology interactions in Sanskrit. The second new point is that Pāṇini *could* not have done for reasons internal to his system.

I shall cite two pieces of phonological evidence that phonology does apply cyclically within words in Sanskrit. The first comes from the accentuation of words with multiple accents. The tradition points out that an accent which is assigned supersedes accents which have been assigned (**satiśiṣṭasvaro balīyān bhavati**, Patañjali on 6.1.158 vt. 8). That is, the last suffix to be added determines the accent of the whole word. This generalization comes for free if rules are applied cyclically from innermost constituents outward.

For example, from the name *Dakṣa*, rule 4.1.95 *ata iñ* yields *Dā́kṣi* 'a descendant of *Dakṣa*' (from *dakṣa+iÑ*), with initial accent by rule 6.1.197 **ñnity ādir nityam**, which accents the initial syllable of a word having a suffix with diacritic Ñ or N. This in turn yields by rule 4.1.101 **yañiñoś ca** the designation of a remote descendant, *Dākṣāyaṇá* 'great-grandson of *Dákṣa*'. Here rule 6.1.165 **kitaḥ** puts the accent on the last syllable, overriding all accents that have been previously assigned in the course of the derivation. As a cyclic derivation would predict, the accent assigned by the last suffix wins.

Consider the derivation of *kurutáḥ* 'they (Du.) make'. If we assume that the order of suffixes matches the derivational sequence in which they are added, then the the cyclic principle predicts the following derivation (with irrelevant steps omitted):

(157)	Tense assignment:	(111) 3.2.123 **vartamāne laṭ**	kṛ+LAṬ
	Root accent:	3.1.91 **dhātoḥ**	kṛ́+LAṬ
	Vikarana placement:	3.1.79 **tanādikṛñbhya uḥ**	kṛ́+u+LAṬ
	Suffix accent:	(12) 3.1.3 **ādyudāttaś ca**	kṛ+ú+LAṬ
	Inflection:	(66) 1.4.108 **śeṣe prathamaḥ**	kṛ+ú+tas
	Suffix accent:	(12) 3.1.3 **ādyudāttaḥ**	kṛ+u+tás

Further rules would then give the correct *kurutáḥ*.

This is, in fact, *not* the Pāṇinian derivation. In his system, for theory-internal reasons, the *vikaraṇas* are added after the person/number endings. In this case, the generalization that the last-added suffix wins (that is, **satiśiṣṭasvaro balīyān bhavati**) predicts the wrong accentuation:

(158)	Tense assignment:	(111) 3.2.123 **vartamāne laṭ**	kṛ+LAT
	Root accent:	3.1.91 **dhātoḥ**	kṛ́+LAT
	Agreement:	(66) 1.4.108 **śeṣe prathamaḥ**	kṛ+tas
	Suffix accent:	(12) 3.1.3 **ādyudāttaś ca**	kṛ+tás
	Vikarana placement:	3.1.79 **tanādikṛñbhya uḥ**	kṛ+u+tás
	Suffix accent:	(12) 3.1.3 **ādyudāttaś ca**	*kṛ+ú+tas

which ultimately gives **kurútaḥ*. Under these assumptions about the morphology, the cyclic principle does not work.

To get the correct form *kurutáḥ,* the traditional interpretation adds an *ad hoc* exception to the general principle that an assigned accent supersedes earlier accents: namely, that *vikaraṇa* accents (such as that on *u* in the above derivation) do not supersede previously assigned accents on personal endings (**satiśiṣṭavikaraṇasvaro lasārvadhātukasvaraṃ na bādhate**, Pat. ad 6.1.158 vt. 10). Pāṇini's treatment of the verb morphology thus complicates the assignment of word accent. If (unlike Pāṇini) we assume that suffixes are always added to the end, so that their linear order in the word matches the derivational order of affixation, then cyclic application of phonological rules gives the right results even in this case.

A second set of cases where Sanskrit shows cyclic application of phonological rules that the grammar does not capture comes from the behavior of roots that have both prefixes and suffixes. The morphology of inflexional suffixes is sensitive to whether the root is prefixed or not. Several suffixes show allomorphic variation based on this, notably the absolutive (gerund) suffix, which makes temporal adverbials with the meaning 'having V-ed', 'after V-ing'. Recall from (147) 7.1.37 **samāse 'nañpūrve ktvo lyap** that it has two basic allomorphs, *-tvā*, which occurs after simple roots, and *-ya*, which occurs after prefixed roots. The latter allomorph gets a *t* added before it if the root is light, in order to make the one-mora root syllable into a minimal foot ((148) 6.1.71 **hrasvasya piti kṛti tuk**). The allomorphy is illustrated by the simple form and a compounded form of the root /bhṛ/ 'carry' in (159).

(159)	a. *bhṛ-tvā́*	'having brought'	(*-tvā* after a simple root)
	b. *saṃ-bhṛ́-tya*	'having brought together'	(*-t-ya* after a light prefixed root)

In *bhṛ-tvā́*, the root, being simple, selects the allomorph *-tvā́*. In (159b) *saṃ-bhṛ́-tya*, the prefixed root selects the absolutive allomorph *-(t)ya*. This shows that the absolutive is formed off the prefixed root.[8]

Additional evidence is the special behavior of the negative prefix *a-*. Unlike verbal prefixes, such as *sam-* in (159b), *a-* has no effect on the choice of absolutive allomorph. For example, *á-bhṛ-tvā* in (159c) has the absolutive allomorph that is otherwise selected by simple roots. Why does *a-* differ from the verbal prefixes in this way? The solution to this puzzle is that it is prefixed not to roots but to absolutives. So, if those absolutives are formed from simple roots, they will have

[8] More precisely, it shows it provided we agree that that the right allomorph is selected at the point at which the morphological operation introducing the affix takes place, and that (contrary to Pāṇini) there are no "allomorphy rules" that could, for example, replace *-tvā́* by *-tya* after the prefix has been added.

the allomorph -*tvā*. The reason why *a*- *must* be prefixed to absolutives and not to roots is that *a*- selects nominal and adverbial stems, and the absolutive suffix -*tvā* makes verbs into adverbs (with essentially nominal character). Conversely, verbal prefixes must be added to roots prior to absolutive formation because they select verbal stems, and absolutives, not being verbs, do not satisfy that subcategorization requirement. The allomorphy contrast between (160a) and (160b) reflects this intrinsic difference in derivational history.

(160) a. saṃ-bhṛ → (saṃ) (bhṛ́-tya) (suffixation to a prefixed root)
b. bhṛ-tvā́ → (á) (bhṛ-tvā) (prefixation to a suffixed stem)

The prosodic structure of the words is the same as far as we can tell, as indicated in (160). Specifically, phonology shows that there is a compound boundary between the prefix and the root in both words. Examples like this show that level ordering cannot be simply reduced to the domains defined by prosodic structure. Rather, the morphophonology reveals two different orders of prefixation and suffixation, as determined by the different selectional requirements of the prefixes and suffixes, for what surfaces as the same prosodic structure.

Because Pāṇini does not adopt the cyclicity of word phonology, he has to stipulate in (147) that the negative prefix *a*- does not trigger the absolutive allomorph -*ya* like other compound roots do.

Another argument for the prefix+root constituent is that prefixed roots can be suffixed with the agent suffix -*tṛ*, which otherwise is not allowed in compounds. Nouns in -*tṛ* are subject to the constraint that they are not compounded (2.2.16 **kartari ca**). If prefixation creates compound roots, then they can be inputs to the affixation of -*tṛ*, and prefixes need not be exceptions to the ban on compounding.

Certain phonological rules also apply cyclically to the prefixed root prior to further affixation. After prefixes ending in -*i* and -*u*, a root-initial *s*- becomes retroflexed to *ṣ*. For example, the root *svaj* 'embrace' (as in *svajate* 'embraces') appears as -*ṣvaj* after the prefix *pari*-, as in *pariṣvajati*. Crucially, this happens even if an augment or a reduplication intervenes (8.3.63-64 ff.), as in the imperfect /pari-a-svaj-a-t/ *paryaṣvajat*, and in the perfect /pari-sa-svaj-e/ *pariṣaṣvaje*.

Importantly, it is *not* possible to account for the "overapplication" of retroflexion on the basis of the output form. The rule that effects this retroflexion otherwise requires strict adjacency between the triggering high vowel and the undergoing *s*. The cyclic nature of the effect is shown even more clearly by cases like *abhi-ta-ṣṭhau*. The root *sthā* 'stand' is prefixed and its initial ṣ- is retroflexed. The following plosive *th* is not retroflexed at this point because assimilation of retroflexion is postlexical. The cyclic derivation, however, yields the correct output form: *sthā*- → *abhi-ṣthā*- → (formation of perfect stem) *abhi-ta-ṣthā*- → (inflection) *abhi-ta-ṣthau* → (postlexical phonology) *abhi-ta-ṣṭhau*.

Because Pāṇini does not adopt the interleaving of phonology and morphology inside words, this derivation is not available to him. He simply stipulates that the augment and the retroflexion may intervene between the high vowel and the *s*.

Why did Pāṇini not adopt word-internal cyclicity? The most important reason is that he treated allomorphy as replacement. This forced him to prevent

the underlying allomorph from triggering unwanted applications of phonological rules prior to being replaced by the actual derived allomorph.

There are many instances where the *antaraṅga*-principle, if applied word-internally, would give the wrong result, and where Pāṇini did not intend it to apply. Instead, he relied on the *siddha*-principle, which, in fact, works correctly in these cases. A typical example is the following.

In the derivation of *seduṣas* (Sg.Gen. of *sed-vas*, Nom. *sedivān*) 'having sat', the suffix+root combination *sed-vas* is subject to a rule which inserts the augment *i-* before consonantal endings. The semivowel *v* of the suffix is then vocalized before accented vocalic endings, which bleeds the insertion of the augment. Cyclicity predicts the incorrect form:

(161) sed-vas
sed-ivas 7.2.35 **ārdhadhātukasyeḍ valādeḥ**
sed-ivas-as 2.3.50 **ṣaṣṭhī śeṣe**
sed-iuas-as 6.4.131 **vasoḥ samprasāraṇam**
sed-ius-as 6.1.108 **saṃprasāraṇāc ca**
*sed-yuṣ-as 8.3.59 **ādeśapratyayayoḥ**, (25) 6.1.77 **iko yaṇ aci**

The *siddha*-principle predicts the correct derivation, where 6.4.131 bleeds 7.2.35, as desired:

(162) sed-vas-as
sed-uas-as 6.4.131 **vasoḥ samprasāraṇam**
sed-us-as 6.1.108 **saṃprasāraṇāc ca**
sed-uṣ-as 8.3.59 **ādeśapratyayayoḥ**
— 7.2.35 **ārdhadhātukasyeḍ valādeḥ** (inapplicable)

It is because of such cases, I think, that Pāṇini abandoned the cyclic principle in the word domain favor of exclusive reliance on the *siddha*-principle, at the cost of complications such as those we discussed.

If we look at such examples from the perspective of today's approaches to morphology, we come to the conclusion that they do not involve competition between phonological rules, but competition between a phonological rule and an allomorphy rule. An allomorphy analysis would posit two allomorphs *-ivāns* and *-us*, which would both combine with the root to give two stems {*sed-ivāns-, sed-us-*}. The selection between those two stems would be done by a fairly general constraint which selects the weakest available stem allomorph before a following accented vocalic case suffix. On this account, there is no question of competing processes. Each allomorph is simply subject to the appropriate phonological rules.

To summarize: the fact that Pāṇini did not adopt the principle of cyclic rule application word-internally is deeply connected with his whole approach to the phonology/morphology interface. In fact, it is inevitable once the decision is made to assimilate allomorphy to phonology by treating it by replacement rules (rather than by a selectional mechanism). Within Pāṇini's system, that treatment of allomorphy is very solidly motivated by the need to provide for inheritance of morphological properties between the allomorphs of a morpheme

(as the default case). But the price to be paid for it is that the phonological input representation will always have the underlying form of each morpheme, which may not be the allomorph that appears in the output. Under the cyclic hypothesis embodied in the *antarañga*-principle, it can happen that this "wrong" allomorph triggers unwanted phonological processes in an inner constituent prior to being replaced by the "right" allomorph in an outer constituent. A secondary reason, as explained above, was that his treatment of the *vikaraṇas* as inserted between root and inflectional affix compromised the cyclic explanation for accent dominance.

Bibliography

Bronkhorst, J.: The role of meanings in Pāṇini's grammar. Indian Linguistics 40, 146–157 (1979)

Bronkhorst, J.: Asiddha in the Aṣṭādhyāyī: a misunderstanding among the traditional commentators? Journal of Indian Philosophy 8, 69–85 (1980)

Deo, A.: Derivational morphology in inheritance-based lexica: Insights from Pāṇini. Lingua 117, 175–201 (2007)

Fong, V.: The order of things: what directional locatives denote. Ph.D. Dissertation, Stanford University (1997)

Houben, J.: 'Meaning statements' in Pāṇini's grammar: on the purpose and context of the Aṣṭādhyāyī. Studien zur Indologie und Iranistik 22, 23–54 (1999)

Joshi, S.D.: The functions of asiddhatva and sthānivadbhāva in Pāṇini's Aṣṭādhyāyī. C.A.S.S. Studies 6, 153–168 (1982)

Joshi, S.D., Bhate, S.: Fundamentals of Anuvṛtti. Poona University Press, Poona (1984)

Joshi, S.D., Kiparsky, P.: Siddha and asiddha in Paninian Phonology. In: Dinnsen, D. (ed.) Current Approaches to Phonological Theory. IU Press, Bloomington (1979)

Joshi, S.D., Roodbergen, J.A.F.: Patañjali's Vyākaraṇa-Mahābhāṣya, Kārakāhnika. University of Poona, Poona (1980)

Kiparsky, P., Staal, F.: Syntactic and semantic relations in Pāṇini. Foundations of Language 5, 83–117 (1969)

Kiparsky, P.: Pāṇini as a Variationist. Poona University Press/MIT Press, Poona/Cambridge (1979)

Kiparsky, P.: Some Theoretical Problems in Panini's Grammar. Professor K. V. Abhyankar Memorial Lectures, Second Series. Post-graduate and Research Department Series No. 16, Poona (1982)

Ostler, N.: Case-Linking: a theory of case and verb diathesis applied to Classical Sanskrit. PhD Thesis, MIT (1979)

Subrahmanyam, P.S.: Pāṇinian linguistics. Tokyo (1999)

Stump, G.: Inflectional morphology: a theory of paradigm structure. Cambridge University Press, Cambridge (2001)

Talmy, L.: Force dynamics in language and cognition. Cognitive Science 12, 49–100 (1988)

Modeling Pāṇinian Grammar

Peter M. Scharf

Department of Classics, Brown University, PO Box 1856, Providence, RI 02912
scharf@brown.edu

Abstract. The current paper compares obvious methods to implement a few aspects of Sanskrit grammar computationally, comments upon the degree to which they approach or depart from Pāṇinian methodology and exemplifies methods that would achieve a closer model. Two questions essential to determining a basic framework in which to implement Pāṇinian grammar computationally are dealt with in some detail: the question of levels and the role of semantics. Pāṇini does not operate with a fourfold hierarchy of modular levels that segregates semantics, syntax, morphology, and phonetics. Rather he conceives of two levels, meaning and sound, generating the latter from the former. He achieves the complex mapping of the former onto the latter utilizing a number of stages that do not correspond neatly to the four modules articulated in modern generative grammar. Although Pāṇini does not state semantic rules, he does operate with numerous semantic categories and sometimes utilizes morphophonemic categories to determine such categories.

Keywords: levels, generative grammar, Panini, Patanjali, Astadhyayi, sandhi, morphology, inflection, syntax, semantics, morphophonemic, syntacticosemantic, circularity, mutual dependence, computational implementation.

Introduction

It is possible to achieve the implementation of generative grammars and parsers of Sanskrit using various methodologies which have varying degrees of affinity to those of Pāṇinian grammar. The current paper compares obvious methods to implement a few aspects of Sanskrit grammar computationally, comments upon the degree to which they approach or depart from Pāṇinian methodology and exemplifies methods that would achieve a closer model. Two questions essential to determining a basic framework in which to implement Pāṇinian grammar computationally are dealt with in some detail: the question of levels and the role of semantics.

1 Differences among Sanskrit Grammarians and Even Pāṇinians

In attempting to create a computational model of Pāṇinian grammar, the first problem is to determine which Pāṇinian grammar. The *Aṣṭādhyāyī* itself (c. 500 B.C.E.), consisting of nearly 4,000 rules, is known to have undergone modifications. Kātyāyana's

G. Huet, A. Kulkarni, and P. Scharf (Eds.): Sanskrit CL 2007/2008, LNAI 5402, pp. 95–126, 2009.

approximately 4,300 vārtikas (4th-3rd c. B.C.E.) suggest modifications to 1,245 of Pāṇini's rules, usually in the form of additions (*upasaṅkhyāna*). Patañjali's *Mahābhāṣya* (mid-2nd c. B.C.E.) rejects many additions suggested by Kātyāyana, suggests other desiderata (*iṣṭi*), and articulates principles presupposed in the grammar. Many of the modifications Kātyāyana and Patañjali suggest are found adopted in the form in which the rules are found in Jayāditya and Vāmana's *Kāśikā*, the oldest extant complete running commentary on the *Aṣṭādhyāyī* (7th c. C.E.). Does one wish to model the *Aṣṭādhyāyī* alone? The *Aṣṭādhyāyī* and Kātyāyana's vārtikas? The grammar as known and approved by Patañjali in the *Mahābhāṣya*? Or the grammar as found in the *Kāśikā*?

2 Ambiguities in Early Articulations Explicated Differently by Subsequent Indian Linguists

Articulations of Pāṇinian grammar, especially sūtras and vārtikas isolated from commentary, are subject to ambiguities. These ambiguities are resolved in different ways by different commentators. Commentaries on Patañjali's *Mahābhāṣya* disagree with each other; commentaries on the *Kāśikā* disagree with each other; and Bhaṭṭojidīkṣita's *Siddhāntakaumudī* (17th c. C.E.) differs in its interpretation of rules and procedures from Jayāditya and Vāmana's *Kāśikā*. Moreover, subcommentaries differ in their interpretations. One must determine the manner in which these ambiguities are to be resolved. Are they to be resolved using some particular commentator as the authority? Haphazardly? Or is one going to come to an independent judgment of the correct interpretation after a critical evaluation of the various interpretations?

Moreover, the supplements to the grammar (see Fig. 1), particularly the lists (*gaṇa*) referred to in various rules, most prominently the list of roots, *Dhātupāṭha*, have undergone variation. Three complete commentaries composed in Sanskrit are extant on the Pāṇinian *Dhātupāṭha*, which is known only through these commentaries: the *Kṣīrataraṅginī* of Kṣīrasvāmin (early twelfth century C.E. Kashmir), the *Dhātupradīpa* of Maitreyarakṣita (mid-twelfth century C.E. Bengal), and the *Mādhavīyadhātuvṛtti* of Sāyaṇa (fourteenth century C.E. Vijayanagara, Karnataka). Will one use one of these? A unified critical edition of them? Or will one attempt to reconstruct the *Dhātupāṭha* as known to Patañjali? Other lists (*gaṇa*) are specified only in commentaries, and many of these are called paradigmatic rather than exhaustive. Will one rely on lexical lists external to the grammar, such as nighaṇṭus and kośas, to complete these lists? (see Fig. 2)

Before embarking on a computational implementation of Pāṇinian grammar, such decisions ought to be made. It may prove very interesting to compare computational implementations based upon different rule sets, different interpretations, and different

sets of supplementary lists with each other and with different sets of linguistic data. As I have argued in two papers, with respect to the derivation of subjunctives (2005, 2008) and of the present stems of class eight roots (forthcoming), systematic comparison of linguistic descriptions resulting from computational implementations with each other and with various collections of extant Sanskrit texts may throw important light upon interpretational and historical questions.

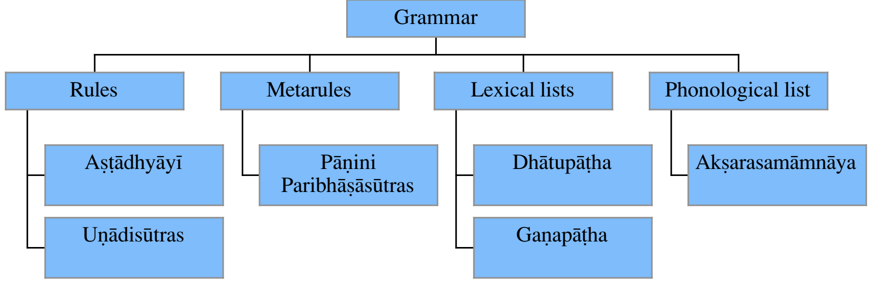


Fig. 1. Grammar Components

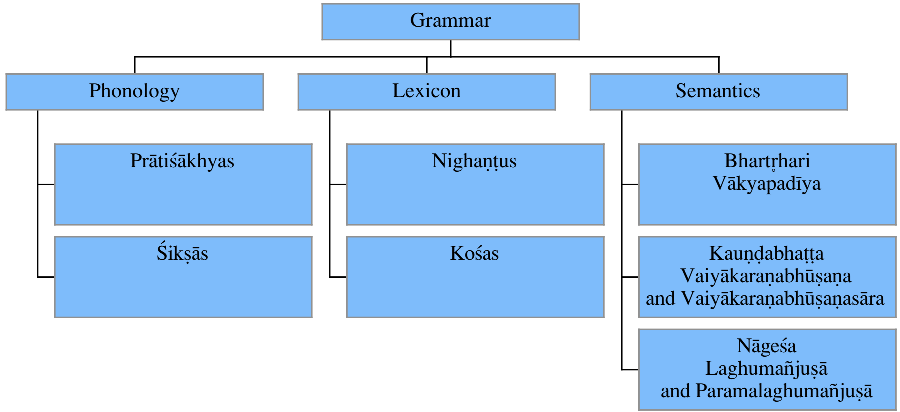


Fig. 2. Implicit Grammar Components

3 Utilization of Contemporary Linguistic Models, in Particular Those Derived from Pāṇinian Methodology, to Articulate Pāṇinian Methodology

Indian grammatical commentaries composed in Sanskrit over the last two and half millennia are not the only sources of Pāṇinian interpretation. Recent work in theoretical and computational linguistics has influenced the interpretation of Pāṇinian grammar.

3.1 Influence of Pāṇinian Methodology on Contemporary Linguistics Generally

Although often not explicitly acknowledged by the influential linguists indebted to it nor recognized by historians of linguistics, Pāṇinian grammar has had a profound influence on modern linguistics. Apart from the influence of ancient Indian phonology on modern phonetic feature analysis, and the emulation of ancient Indian synchronic sound change laws by diachronic laws of phonological change in modern historical and comparative linguistics, Pāṇinian grammar supplied the basic archetype at the foundation of modern generative grammar. From Chomsky's first work on transformational grammar in 1957 to the Pāṇinian grammars of modern Indian languages such as described for Hindi in Bharati et al 1995, modern linguistic science is heavily indebted to the concepts and procedures of ancient Indian linguistics.

3.2 Influence of Contemporary Linguistic Models on the Interpretation of Pāṇinian Methodology

Concepts originally inspired by ancient Indian linguistics have taken their own shape in contemporary linguistics. They have responded to different concerns and been adapted to different questions. These new concepts have been applied by contemporary scholars to the interpretation of Pāṇinian grammar. One of the most prominent of these is the idea that grammar consists of modules in a generative hierarchy, or levels.

4 Levels

4.1 Kiparsky's Architecture

Clearly influenced by Chomskian generative grammar, Kiparsky and Staal (1969) proposed that Pāṇinian grammar contains rules in a hierarchy of four levels of representation: semantics, deep structure, surface structure, and phonology. More recently Kiparsky (2002) restates this scheme referring to the four levels as follows: (1) semantic, (2) morphosyntactic, (3) abstract morphological, and (4) phonological (see Fig. 3). Three classes of rules map prior levels onto subsequent levels: (1) rules that assign kārakas and abstract tense, (2) morphological spellout rules, and (3) rules of allomorphy and phonology. Rules incorporate conditions at both the levels from which and to which they map, as well as at prior levels in a unidirectional derivation beginning with semantics and ending with phonology.

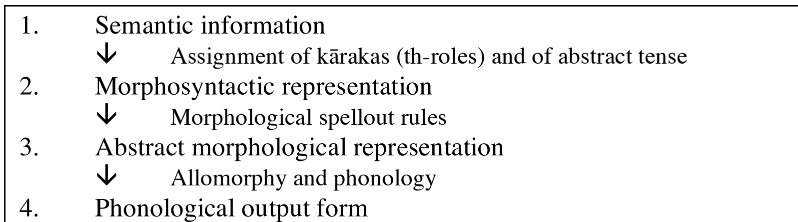


Fig. 3. Levels according to Kiparsky 2002: 3

As an example of how derivation is understood to work in the four-level hierarchy, one may take the derivation of the sentence *devadatta odanaṁ pacati* (Fig. 4). At the semantic level, the speaker intends to express that Devadatta, called here John Doe, undertakes the action of cooking in present time for the purpose of making boiled rice. Pāṇinian semantics classifies John Doe as the independent agent in the action, and boiled rice as that which is desired to be obtained. Three rules apply to map the semantic level onto the morphosyntactic level. 1.4.49 and 1.4.54 assign kārakas, and 3.2.123 assigns abstract tense by introducing the *l*-affix *laṭ* on the condition that present time is to be denoted.

1. John Doe$_{\text{[svatantra]}}$ rice$_{\text{[īpsitatama]}}$ cooks$_{\text{[vartamāna]}}$.
   John Doe$_{\text{[independent]}}$ rice$_{\text{[desideratum]}}$ cooks$_{\text{[present]}}$.

   1.4.49 *karturīpsitatamaṁ karma*
   ↓ 1.4.54 *svatantraḥ kartā*
   3.2.123 *vartamāne laṭ*

2. Devadatta$_{\text{[kartṛ]}}$ odana$_{\text{[karman]}}$ **ḍu**pac**aṣ**+l**aṭ**.
   Devadatta$_{\text{[agent]}}$ odana$_{\text{[direct object]}}$ pac+l**aṭ**.

   3.4.78 *tiptasjhi...iḍvahimahiṅ*
   1.3.78 *śeṣātkartari parasmaipadam*
   1.4.108 *śeṣe prathamaḥ*
   1.4.22 *dvyekayor dvivacanaikavacane*
   ↓ 3.1.68 *kartari śap*
   4.1.2 *svaujasamauṭ...ṅyossup*
   2.3.2 *karmaṇi dvitīyā*
   2.3.46 *prātipadikārthaliṅgaparimāṇavacanamātre prathamā*

3. Devadatta+s**u** odana+am **ḍu**pac**aṣ**+**ś**a**p**+ti**p**.
   Devadatta+$_{\text{[nom]}}$ odana+$_{\text{[acc]}}$ pac+$_{\text{[3sa pre]}}$.

   1.3.9 *tasya lopaḥ*
   6.1.107 *ami pūrvaḥ*
   ↓ 8.3.17 *bhobhagoaghoapūrvasya yo 'śi*
   8.3.19 *lopaḥ śākalyasya*
   8.3.23 *mo 'nusvāraḥ*

4. Devadatta odanaṁ pacati.
   Devadatta cooks rice.

Fig. 4. Example of Four-level Derivation

Several "spellout" rules then apply to map the morphosyntactic level onto the abstract morphological level. 3.4.78 provides that basic verbal terminations replace the *l* of the affix *laṭ* that occurs after the verbal root *pac*. Restrictive rules 1.3.78, 1.4.108

and 1.4.22, read in conjunction with 3.4.78, select the third person singular active (3sa) affix *tip* on condition that a single agent that is neither the speaker nor the addressee is to be denoted. Before the affix *tip* (termed *sārvadhātuka* by 3.4.113 *tiṅśit sārvadhātukam*), 3.1.68 provides the default verbal stem-forming affix *śap* to cosignify the agent. Then 4.1.2 provides nominal terminations. Restrictive rules 2.3.2, 2.3.46, and 1.4.22, read in conjunction with 4.1.2 select the appropriate nominal termination. 2.3.2 selects a second triplet nominal termination (*dvitīyā*) after the stem *odana* on condition that the kāraka *karman*, which has not yet been denoted (*anabhihite* 2.3.1), is to be denoted. 2.3.46 selects a first triplet nominal termination (*prathamā*) after the stem *devadatta* on condition that just the stem meaning, gender, and number are to be denoted. (The kāraka *kartṛ* has already been denoted by the verbal termination thus preventing 2.3.18 *kartṛkaraṇayos tṛtīyā* from applying.) 1.4.22 selects the singular terminations *am* (2s) and *su* (1s), respectively in each triplet.[1]

Finally, several rules of allomorphy (of which there are none in the present example) and phonology apply to map the abstract morphological level onto the phonological level.[2]

4.2 Houben 1999

Houben (1999) aptly criticized earlier articulations of this four-level hierarchy because they did not explicitly include pragmatics and intentionality in the semantic level and did not permit semantic factors (including pragmatics and intentionality) to serve as conditions in phonological rules directly. Figure 5 shows Houben's (1999: 46) model of the four-level hierarchy. In addition, he criticized the portrayal of Pāṇini's grammar as a complete automaton that produces utterances from meanings.

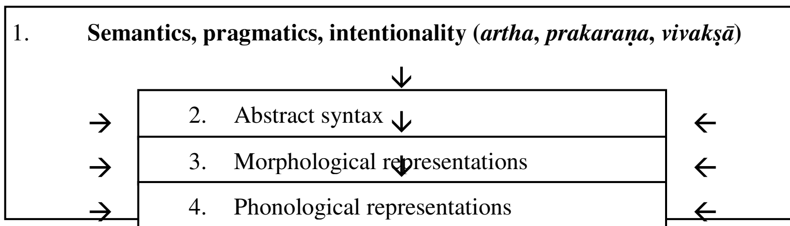


Fig. 5. Levels according to Houben 1999: 46

[1] Rules 1.4.99-108 that designate verbal and nominal terminations in the lists 3.4.78 and 4.1.2 by terms that allow selection according to person, number, and voice are not shown.

[2] The rule that deletes markers, 1.3.9, is shown here though its application is simultaneous with the introduction of affixes.

He pointed out that there are no rules that introduce verbal roots and nominal stems based upon semantic conditions and that the fundamental morphophonemic elements appear in Pāṇinian derivations from the start. It is therefore improper, he argued, to characterize the grammar as originating in semantics and culminating in phonological form. Rather, he (1999: 48) stated, it originates in meaning mixed with form and culminates in a perfected form.

4.3 The Purpose of the Science of Language

Houben is correct to reemphasize that it is not the function of the *Aṣṭādhyāyī* to teach semantics. The science of grammar does not teach the communication of meaning that is already known from ordinary usage; rather, it teaches correct usage in the conveyance of the desired meaning. In his very first vārtika, commented upon at length by Patañjali in the *Paspaśāhnika*, Kātyāyana places the function of grammar in the context of what is already known from ordinary behavior. There is an established relation between words and their objects, which is known from ordinary usage, such that certain words are used to denote certain objects. The purpose of using speech forms is to convey knowledge of objects by following the conventions of ordinary usage. Since this is the case, the purpose served by the science of grammar is to make known which speech forms among those in use are correct and hence lead to dharma. Kātyāyana states:

> *Siddhe śabdārthasambandhe lokato 'rthaprayukte śabdaprayoge śāstreṇa dharmaniyamaḥ, yathā laukikavaidikeṣu.*[3]
> Since speech, its object, and the relation between the two are established (and are known) from ordinary usage, and since one uses speech prompted by meanings in accordance with ordinary usage, the science (of grammar) restricts (usage to correct speech forms) for the sake of dharma just as (other disciplines restrict behavior) in ordinary and Vedic affairs.[4]

4.4 Semantics

While it is obviously correct that the *Aṣṭādhyāyī* does not include any rules that are concerned with semantics to the exclusion of syntax, morphology, and phonology, the system of rules clearly presupposes that semantics drive the derivation. In normal communication, meaning is the reason for speech. Under 1.1.44, Patañjali describes that the purpose of speech is to convey understanding:

> The use of words is for the purpose of the comprehension of the objects they denote. With the intention, "I will give the understanding of an object" a word is used.[5]

---

[3] K1.6.8.
[4] Scharf 1995.
[5] *Arthagatyarthaḥ śabdaprayogaḥ. Arthaṁ saṁpratyāyayiṣyāmīti śabdaḥ prayujyate.* K1.105.2.

Modeling the fact that a speaker selects speech forms to use on the basis of the meaning he wishes to convey, the *Aṣṭādhyāyī* is composed in a manner that selects certain speech forms for use on the basis of certain semantic conditions. Specific semantic factors pervasively serve as conditions for the classification of lexical items, and for the introduction of kāraka terms, cover symbols (abstract symbols that stand for subsequent phonological replacements), and speech forms.

4.4.1 Lexical Organization

The use of words in rules to refer to classes of words rather than just to their own speech form is discussed in the *Mahābhāṣya* under 1.1.68 *svaṁ rūpaṁ śabdasyāśabdasañjñā*. Table 1 summarizes the various modes of reference along with examples of each. The word *agni* 'fire' in 4.2.33 *agner ḍhak* refers to the speech form *agni* itself in accordance with the general principle stated in 1.1.68, not to its meaning. However, departing from the general principle, the word *vṛkṣa* 'tree', etc. in 2.4.12 *vibhāṣā vṛkṣamṛga...* refers to terms for species of trees.[6] The word *sva* 'property', etc. in 3.4.40 *sve puṣaḥ* refers to itself as well as to its synonyms,[7] while the word *rājan* in 2.4.23 *sabhā rājāmanuṣyapūrvā* refers to its synonyms but not to itself. Finally, the word *matsya* in 4.4.35 *pakṣimatsyamṛgān hanti* refers to itself as well as to terms for species of fish. The use of words in the grammar to refer to classes of words, rather than to the speech forms themselves, succeeds through the intermediary of the words' meaning; this use contrasts with the norm in the grammar for words to refer just to their own form. By referring to their meaning, in the way words are ordinarily used, the meaning of the word can serve as the condition to class groups of words of related meaning.

Table 1. Modes of lexical reference

- Speech forms in the *Aṣṭādhyāyī* generally refer to themselves:
  4.2.33 *agner ḍhak*
- But some lexemes refer to the members of a class they denote:
  2.4.12 *vibhāṣā vṛkṣamṛgatṛṇadhānyavyañjanapaśuśakunyaśva-vaḍavapūrvāparādharottarāṇām*
- Some refer to their synonyms as well as themselves:
  3.4.40 *sve puṣaḥ*
- Some refer to their synonyms rather than to themselves:
  2.4.23 *sabhā rājāmanuṣyapūrvā*
- Some refer to the members of a class they denote as well as to themselves:
  4.4.35 *pakṣimatsyamṛgān hanti*

---

[6] *sit tadviśeṣāṇāṁ vṛkṣādyartham.* vt. 5, K I.176.25. The scheme of distinguishing the ways in which words are used to refer to various classes of words or to themselves proposed in vārtikas 5-8 is not adopted in the *Aṣṭādhyāyī*. It nevertheless illustrates these various usages in the grammatical treatise.

[7] *pit paryāyavacanasya ca svādyartham.*

Table 2. Examples of semantic conditions in the locative

• *deśe*		3.3.78, 4.2.52, 4.2.67, 4.2.119, 5.2.105, 5.2.135, 6.3.98, 8.4.9
• *adeśe*		8.4.24
• *janapade*		4.2.81
• *janapadatadavadhyoḥ*		4.2.124
• *nadyām*		4.2.85
• *parvate*		4.3.91
• *parimāṇe*		4.3.153, 5.2.39
• *jātau*	genus (*jāti*)	4.1.161, 5.2.133
	non-genus (*ajāti*)	5.4.37, 6.4.171
	species (*jāti*)	6.3.103
	ethnicity (*jāti*)	6.2.10
• *vayasi*		3.2.10, 4.1.20, 5.1.81, 5.2.130, 5.4.141, 6.2.95
• *avayasi*		5.1.84
• *matsye*		5.4.16
• *cittavati*		5.1.89

There are 735 words used in the locative to state semantic conditions in rules (including repetitions and excluding individual compound elements). Table 2 shows several examples. Conditions that serve to classify lexical items include place (*deśa*);[8] district (*janapada*);[9] river (*nadī*);[10] mountain (*parvata*);[11] measure (*parimāṇa*);[12] genus,[13] species,[14] or ethnicity (*jāti*);[15] age (*vayas*);[16] fish (*matsya*); and conscious being (*cittavat*);[17] among others.

4.4.2 Semantic Conditions for kārakas, Cover Symbols, and Phonetics

It is well known that the terms *dhruva* 'fixed point', etc. in rules 1.4.24-55 *dhruvamapāye 'pādānam,* etc., shown in Table 3, serve as semantic conditions for the introduction of kāraka terms, and that terms such as *bhūta* 'past', *vartamāna* 'present', and *bhaviṣyat* 'future', used in the locative in 3.2.84 *bhūte*, 3.2.123 *vartamāne laṭ*, and 3.3.3 *bhaviṣyati gamyādayaḥ*, as shown in Table 4, serve to introduce *l*-affixes. Houben (1999: 46) has illustrated the direct use of semantic and pragmatic factors as conditions for phonetic modifications to strings in the section of rules 8.2.82-108 (see Table 5). Such factors conjoin with the syntactic condition, specified in the heading to the section, 8.2.82 *vākyasya ṭeḥ pluta udāttaḥ*, that the string be a sentence (*vākya*).

---

[8] *deśa* 3.3.78, 4.2.52, 4.2.67, 4.2.119, 5.2.105, 5.2.135, 6.3.98, 8.4.9; *adeśa* 8.4.24.
[9] 4.2.81, 4.2.124.
[10] 4.2.85.
[11] 4.3.91.
[12] 4.3.153, 5.2.39.
[13] *jāti* 4.1.161, 5.2.133; *ajāti* 5.4.37, 6.4.171.
[14] 6.3.103.
[15] 6.2.10.
[16] *vayas* 3.2.10, 4.1.20, 5.1.81, 5.2.130, 5.4.141, 6.2.95; *avayas* 5.1.84.
[17] 5.1.89.

Table 3. Semantic conditions for kāraka classification

sūtra	kāraka term	semantic condition
1.4.24	*apādāna*	fixed point of departure
1.4.32	*saṁpradāna*	intended recipient of the object
1.4.42	*karaṇa*	immediately most efficacious
1.4.45	*adhikaraṇa*	substrate
1.4.49	*karman*	most desired to be attained
1.4.54	*kartṛ*	independent

Table 4. Semantic conditions for *l*-affixes

3.2.84	*bhūte*		
		3.2.110	*luṅ*
3.2.123	*vartamāne laṭ*		
3.3.3	*bhaviṣyati gamyādayaḥ*		
		3.3.13	*lṛṭ śeṣe ca*

Table 5. Semantic conditions for phonetics

8.2.82	*vākyasya ṭeḥ pluta udāttaḥ*
8.2.83	*pratyabhivāde 'śūdre*
8.2.84	*dūrāddhūte ca*
etc.	

4.4.3 x-vacana

A number of rules explicitly use the term *vacana* 'denoting' to designate the semantic conditions that serve as the criteria to class together words that denote entities in major categories. Hence, as shown in Table 6, semantic conditions serve to form a class of words that denote entities other than substances (*asattvavacana*),[18] a class of words that denote qualities (*guṇavacana*),[19] a class of words that denote common properties (*sāmānyavacana*),[20] or distinguishing properties (*viśeṣavacana*),[21] and a class of words that denote the essence (*bhāva*) of what is denoted by the stem after which certain affixes forming such words occur (*bhāvavacana*).[22]

Table 6. Semantic conditions designated with the term *vacana*

asattva-vacana	2.3.33, etc.
guṇa-vacana	2.1.30, etc.
sāmānya-vacana	3.4.5, etc.
viśeṣa-vacana	8.1.74, etc.
bhāva-vacana	2.3.15, etc.

Similarly, other rules explicitly use the term *vacana* to designate the semantic conditions that serve as the criteria to form narrower classes of lexemes subject to com-

[18] 2.3.33 *karaṇe ca stokālpakṛcchrakatipayasyāsattvavacanasya.*

[19] 2.1.30 *tṛtīyā tatkṛtārthena guṇavacanena*, 4.1.44 *voto guṇavacanāt*, 5.1.124 *guṇavacanabrāhmaṇādibhyaḥ karmaṇi ca*, 5.3.58 *ajādī guṇavacanād eva*, 6.2.24 *vispaṣṭādīni guṇavacaneṣu*, 8.1.12 *prakāre guṇavacanasya.*

[20] 3.4.5 *samuccaye sāmānyavacanasya*, 8.1.73 *nāmantrite samānādhikaraṇe sāmānyavacanam.*

[21] 8.1.74 *vibhāṣitaṁ viśeṣavacane bahuvacanam.*

[22] The term *bhāvavacana* occurs in three sūtras: 2.3.15 *tumarthāc ca bhāvavacanāt*, 2.3.54 *rujārthānāṁ bhāvavacanānām ajvareḥ*, 3.3.11 *bhāvavacanāś ca*, and the term *bhāvakarmavacana* in one: 6.2.150 *ano bhāvakarmavacanaḥ.*

mon operations. Hence in one rule semantic conditions serve to form classes of indeclinables that denote proximity (*samīpa*), flourishing (*samṛddhi*), lack of prosperity (*vyṛddhi*), absence of an object (*arthābhāva*), going beyond (*atyaya*), unsuitability for the moment (*asaṁprati*), the appearance of a sound or word (*śabdaprādurbhāva*), posteriority (*paścāt*), a meaning of *yathā*, sequence (*ānupūrvya*), simultaneity (*yaugapadya*), similarity (*sādṛśya*), success (*saṁpatti*), completeness (*sākalya*), end (*anta*), and senses denoted by nominal terminations and other affixes provided by rules 5.3.1-26 (*vibhakti*).[23] In other rules the term *vacana* designates classes of words that denote remembrance (*abhijñā*),[24] stages of bodily growth (*vayas*),[25] haste (*kṣipra*),[26] wish (*āśaṁsā*),[27] boundary (*maryādā*),[28] imagination or supposition (*saṁbhāvana*),[29] fitness (*paryāpti*),[30] and half (*sāmi*).[31] In commenting upon several of these rules, the *Kāśikā* notes that the term *vacana* is used to include synonyms of the word that precedes it in compound.[32]

Elsewhere the term *vacana* explicitly designates the semantic condition for a particular triplet of nominal terminations, secondary affix, or finished form (*nipātana*). Such semantic conditions include master (*īśvara*),[33] virgin (*apūrva*),[34] momentary (*ādyanta*),[35] particular sort or manner (*prakāra*),[36] extolled (*prakṛta*),[37] and dependent (*tadadhīna*),[38] The term *vacana* also designates a broad class of semantic conditions that serve as conditions for the formation of tṛtīyā-tatpuruṣa compounds. These include additional significance such as praise or censure (*adhikārtha*).[39]

---

[23] 2.1.6 *avyayaṁ vibhaktisamīpasamṛddhivyṛddhyarthābhāvātyayāsampratiśabdaprādurbhāvapaścādyathānupūrvyayaugapadyasādṛśyasampattisākalyāntavacaneṣu.*

[24] 3.2.112 *abhijñāvacane lṛṭ.*

[25] 3.2.129 *tācchīlyavayovacanaśaktiṣu cānaś*, 5.1.129 *prāṇabhṛjjātivayovacanodgātrādibhyo 'ñ* 6.3.85 *jyotirjanapadarātrinābhināmagotrarūpasthānavarṇavayovacanabandhuṣu.*

[26] 3.3.133 *kṣipravacane lṛṭ.*

[27] 3.3.134 *āśaṁsāvacane liṅ.*

[28] 3.3.136 *bhaviṣyati maryādāvacane 'varasmin*, 8.1.15 *dvandvaṁ rahasyamaryādāvacanavyutkramaṇayajñapātraprayogābhivyaktiṣu.*

[29] 3.3.155 *vibhāṣā dhātau sambhāvanavacane 'yadi.*

[30] 3.4.66 *paryāptivacaneṣv alamartheṣu.*

[31] 5.4.5 *na sāmivacane.*

[32] Under 3.2.112, 3.3.133, 5.4.5 the *Kāśikā* states: *vacanagrahaṇaṁ paryāyārtham.*

[33] 2.3.9 *yasmād adhikaṁ yasya ceśvaravacanaṁ tatra saptamī.*

[34] 4.2.13 *kaumārāpūrvavacane.*

[35] 5.1.114 *ākālikaḍādyantavacane.*

[36] 5.3.23 *prakāravacane thāl*, 5.3.69 *prakāravacane jātīyar*, 5.4.3 *sthūlādibhyaḥ prakāravacane kan.*

[37] 5.4.21 *tatprakṛtavacane mayaṭ.*

[38] 5.4.54 *tadadhīnavacane.*

[39] 2.1.33 *kṛtyair adhikārthavacane.* The *Kāśikā* comments, "The expression of additional meaning is the expression of the superimposed meaning connected with praise or censure." (*stutinindā-prayuktam adhyāropitārtha-vacanam adhikārtha-vacanam*). In 2.3.46 *prātipadikārthaliṅgaparimāṇavacanamātre prathamā*, the term *vacana* is taken by commentators to denote number rather than to refer to reference explicitly, i.e. it is not the case that the rule provides as a condition for the occurrence of a first-triplet nominal termination merely the denotation (*vacana*) of measure (*parimāṇa*), gender (*liṅga*), and the meaning of the stem (*prātipadikārtha*).

4.5 Ontology

In addition to various specific semantic factors that serve as conditions for the classification of lexical items, and for the introduction of kāraka terms, cover symbols, and speech forms, the *Aṣṭādhyāyī* incorporates certain ontological presuppositions. The grammar presupposes a certain structure in the semantic field in order to operate properly. Rules have been formulated with certain conceptions regarding the nature of things in mind. Numerous passages in Patañjali's *Mahābhāṣya* analyze such presuppositions, as do the works of later philosophers of language from Bhartṛhari (5th century C.E.) to Kauṇḍabhaṭṭa and Nāgeśa (seventeenth and eighteenth centuries). Patañjali, for instance, has his interlocutors asks questions concerning the nature of action, time, and change in the course of their arguments about the formulation and scope of rules. They ask:

> What do you consider action to be when you say, "The term *dhātu* doesn't apply to the roots as (class 2), *bhū* (class 1), and *vid* (class 4)."?
> *kāṁ punaḥ kriyāṁ bhavān matvāhāstibhavatividyatīnāṁ dhātusaṁjñā na prāpnotīti.* (1.3.1, vt. 5. K1.258.8-9)

> What do you consider time to be when you say, "The rule doesn't make sense because the object denoted by the word with which the word for time is compounded is not what gets measured."
> *kaṁ punaḥ kālaṁ matvā bhavān āha kālasya yena samāsas tasyāparimāṇitvād anirdeśa iti* (2.2.5, vt. 1. K1.409.21-22)

> What do you consider change to be when you say, "It doesn't work (the taddhita suffix doesn't apply) in the case of *bali* and *ṛṣabha*."?
> *kaṁ punar bhavān vikāraṁ matvāha balyṛṣabhayor na sidhyati.*
> (5.1.13 K2.342.16)

Examination of the *Aṣṭādhyāyī* itself reveals that it presupposes a certain ontology. Substances (*dravya*), qualities (*guṇa*), and actions (*kriyā*) are distinguished as are time (*kāla*), the divisions of time past (*bhūta*), present (*vartamāna*), and future (*bhaviṣyat*), and the degrees of proximity in time near (*āsanna*), today (*adyatana*), and not today (*anadyatana*). Number (*saṁkhyā*) is recognized. Common properties (*sāmānya*) are recognized, as is also essence (*bhāva*). Much of this ontology subsequently appears as categories in the Vaiśeṣeṣika system of philosophy

The various ontological categories refered to in the *Aṣṭādhyāyī* serve as the conditions that characterize sets of speech forms. Speech forms are subject to various operations on the condition that they do or do not denote a certain entity in a certain ontological category. The semantic condition is frequently placed in the locative. For example, 5.4.11 *kimettiṅavyayaghād āmv adravyaprakarṣe* provides a suffix *-ām* (*āmu*) to a stem ending in a comparative and superlative affix *-tara* or *-tama* on the condition that the excellence to be denoted is not located in a substance (*dravya*). Similarly, the speech forms in the list beginning with ca are termed *nipāta* if they do not denote a substance (*sattva*).[40] They are subsequently termed indeclinable

[40] 1.4.57 *cādayo 'sattve.*

(*avyaya*).[41] Other ontological categories that serve as semantic conditions in the locative include time (*kāla*),[42] and essence (*bhāva*).[43]

4.6 Challenges to Unidirectionality

Although the *Aṣṭādhyāyī* does not provide explicit rules exclusively regarding semantics the fact that it does incorporate extensive organization of the semantic field is significant. It is particularly significant that the organization of the semantic field is carried out in part on the basis of reference to syntactic and morphological elements. Such elements are generally introduced subsequently to and on the basis of semantic conditions. Hence, the organization of the semantic field by reference to syntactic and morphological elements challenges the assertion that a hierarchy of levels is unidirectional as asserted by Kiparsky (2002: 3) (see Figure 3).

4.6.1 x-arthe

In the level hierarchy articulated by Kiparsky (2002), Pāṇini employs elements at levels two and three to specify semantic criteria at level one. Twenty-five of the 735 words that specify semantic criteria employ the term *artha* 'meaning' in order to specify semantic conditions on level one on the basis of morphosyntactic elements at level two and morphological elements at level three (see Table 7).[44] In one case, an abstract morphological element on level two is employed to specify a semantic item on level one that serves as a semantic condition for another abstract morphological element at level two. 3.4.7 *liṅarthe leṭ* provides that in Vedic the abstract morphological element *leṭ* occurs in the meaning of the abstract morphological element *liṅ*. In this case, the rule that assigns abstract tense incorporates conditions only at the levels from which and to which it maps; it thereby accords with the general restriction that rules incorporate conditions only at the levels from which and to which they map·

---

[41] 1.1.37 *svarādinipātam avyayam*.

[42] 2.3.64, 5.3.15.

[43] 3.1.107, 3.3.18, 3.3.44, 3.3.75, 3.3.95, 3.3.98, 3.3.114, 3.4.69, 4.4.144, 6.2.25. As a Buddhist, it is natural for Jayāditya to avoid accepting essence as the meaning of the word *bhāva*. Jayāditya understands the root *bhū* to refer to generic action (*kriyāsāmānya*); hence he takes the term *bhāva* to refer to the generic action common to the meaning of any root. In the *Kāśikā* under 3.3.18 *bhāve*, he states *kriyāsāmānyavācī bhavatiḥ*, following Patañjali's statement *kṛbhvastayaḥ kriyāsāmānyavācinaḥ* (K2.144.20, K2.47.24, etc.). Since the affixes provided under the heading of 3.3.18 occur after roots, which denote action, the *bhāvavacana* words refered to in 3.3.11 would denote generic action *kriyāsāmānya* even if the term *bhāva* did refer to essence; the common property in all action is the essence of action. A long tradition of comment on the meaning of the term *bhāva* determines that it denotes static (literally 'non-dynamic') action (*aparispandamāna-kriyā*) when it specifies the condition for nominal affixes. See Rocher 1966.

[44] *saptamyarthe* 1.1.19, *caturthyarthe* 1.3.55, *tṛtīyārthe* 1.4.85, *mātrārthe* 2.1.9, *anyapadārthe* 2.1.21, *cārthe* 2.2.29, *caturthyarthe* 2.3.62, *liṅarthe* 3.4.7, *tumarthe* 3.4.9, *kṛtyārthe* 3.4.14, *matvarthe* 4.4.128, *dhātvarthe* 5.1.118, *vidhārthe* 5.3.42, *jīvikārthe* 5.3.99, *śakyārthe* 6.1.81, *tadarthe* 6.1.82, *nityārthe* 6.2.61, *atadarthe* 6.2.156, *atadarthe* 6.3.53, *īṣadarthe* 6.3.105, *aṇyadarthe* 6.4.60, *śakyārthe* 7.3.68, *upamārthe* 8.2.101, *kṛtvo'rthe* 8.3.43, *adhyarthe* 8.3.51.

Table 7. Semantic conditions designated with the term *artha*

saptamyarthe 1.1.19	*jīvikārthe* 5.3.99
caturthyarthe 1.3.55	*śakyārthe* 6.1.81
tṛtīyārthe 1.4.85	*tadarthe* 6.1.82
mātrārthe 2.1.9	*nityārthe* 6.2.61
anyapadārthe 2.1.21	*atadarthe* 6.2.156
cārthe 2.2.29	*atadarthe* 6.3.53
caturthyarthe 2.3.62	*īṣadarthe* 6.3.105
liṅarthe 3.4.7	*aṇyadarthe* 6.4.60
tumarthe 3.4.9	*śakyārthe* 7.3.68
kṛtyārthe 3.4.14	*upamārthe* 8.2.101
matvarthe 4.4.128	*kṛtvo'rthe* 8.3.43
dhātvarthe 5.1.118	*adhyarthe* 8.3.51.
vidhārthe 5.3.42	

The remaining 25 rules containing words ending in the term *artha* that specify semantic criteria violate the enunciated condition that rules incorporate conditions only at the levels from which and to which they map, as well as at prior levels in the unidirectional hierarchy beginning with semantics and ending with phonology. They incorporate conditions at level three that specify semantic criteria at level one, two levels prior in the unidirectional hierarchy. Two examples suffice to demonstrate the problem. 1.1.19 *īdūtau ca saptamyarthe* provides that the sounds *ī* and *ū* occurring in the meaning of the seventh vibhakti (*Kāśikā*: *saptamyarthe vartamānam*) in the *Padapāṭha* are termed *pragṛhya* and therefore do not undergo sandhi. The rule thereby specifies a semantic element, the meaning of the seventh vibhakti, at level one on the basis of items termed the seventh vibhakti, namely morphological affixes *-ṅi*, *-os*, and *-su*, at level three. The semantic condition in turn specifies a phonological trait, the absence of sandhi, at level four. Similarly, 3.4.9 *tumarthe sesenaseasen...* specifies several affixes that occur in the same meaning as the meaning of the infinitival affix *-tum* (*Kāśikā*: *tumuno 'rthas tumarthaḥ*). The rule thereby employs a morphological element *-tum* at level three to characterize a set of semantic conditions at level one, which then conditions allomorphs *-se*, *-sen*, etc. at level four.

The first example supports the criticism of earlier versions of the levels theory already articulated by Houben (1999) that it did not permit semantic factors to serve as conditions in phonological rules directly. 1.1.19 provides just what was not permitted: the semantic condition consisting of the meaning of the seventh vibhakti inhibits sandhi. The present version of the levels theory accommodates this criticism by permitting rules to incorporate factors at any prior level in the hierarchy as conditions. An additional problem not previously articulated, however, plagues the present version of the levels theory: rules incorporate factors at subsequent levels of the hierarchy as conditions at prior levels.

In Kiparsky's hierarchy of levels, the meaning of the seventh is at a prior level of derivation to the seventh triplet of nominal terminations (*saptamī vibhakti*). One would have to run through the hierarchy to level three to get the seventh triplet terminations in order to establish the semantic range of the meaning of the seventh triplet at level one.

It is not licit to dismiss the problem by claiming that the use of the term *artha* serves merely to state synonymy at levels two or three and does not involve mapping

to the prior semantic level. As Houben (1999) has reiterated, Pāṇini does not state rules that operate exclusively on the semantic level. Yet, as I have demonstrated above, Pāṇini does incorporate organization of the semantic level in his rules. The organization of the semantic level is achieved in part by reference to syntactic and morphological criteria. Since syntactic and morphological criteria serve to express the structure of the semantic level, subsequent levels of the hierarchy, including the morphological level, which is two levels removed, serve as conditions for prior levels.

4.6.2 x-vacana

In two cases of the use of the term *vacana* the semantic condition that serves to characterize a set of speech forms is specified by reference to levels considered to be subsequent to the semantic level in the hierarchy of four levels proposed by Kiparsky and Staal. In 6.2.150 *ano bhāvakarmavacanaḥ*, the kāraka term *karman* designates a class of items that serve as the semantic conditions that characterize a set of speech forms. In accordance with this rule, a subsequent compound element (*uttarapada*) that meets three conditions has its final vowel high-toned. The three conditions are the following: 1. it ends in an affix of the form *ana*; 2. it denotes static action (*bhāva*) or a direct object (*karman*); and 3. it is preceded by a compound element denoting a kāraka. The fact that a kāraka is referred to as the direct object of the root *vac* in the term *vacana* is significant. It indicates that Pāṇini considered kārakas to be denotable just as purely semantic conditions are denotable.

In 2.1.6 *avyayaṁ vibhakti-samīpa-samṛddhi-vyṛddhy-arthābhāvātyayāsamprati-śabdaprādurbhāva-paścād-yathānupūrvya-yaugapadya-sādṛśya-sampatti-sākalyānta-vacaneṣu*, one of the semantic conditions that serves to characterize a class of indeclinables is itself characterized by morphological criteria. The rule provides that indeclinables that occur in a number of senses combine with subsequent elements to form avyayībhāva compounds. The senses specified include those denoted by nominal terminations and other affixes provided by rules 5.3.1-26 (*vibhakti*). Hence the morphemes that constitute vibhaktis serve to characterize the semantic conditions under which certain indeclinables are used. Morphological criteria therefore serve as the grounds for the organization of semantics which was considered a prior level in the hierarchy proposed by Kiparsky and Staal. Note that the adoption of cyclicity in the formation of the compounds in question does not escape the problem of counterdirectionality in the hierarchical ordering. Regardless of whether rules that generate compounds and their accentuation occur subsequent to rules that generate their compound elements, the indeclinable that constitutes the prior element of the avyayībhāva compound must have access to the morphological level even before the question of its entering into a compound arises. Indeclinables are classed according to semantic criteria that are themselves specified by morphological units.

4.6.3 Avoidance of Circularity

Although the seventh vibhakti arises subsequently to its semantic conditions, yet it can serve as the criterion to characterize its semantic conditions without resulting in circularity much in the way circularity is avoided by invoking the fact that speech is abiding. Indian linguists typically assert the fact that speech is abiding (*siddha* or *nitya*) as opposed to transient (*kārya*). If a speech form were new in each instance of its utterance, it could not form a relation with a meaning and could not convey

meaning. Only recognized by the speech community as the same in each instance of its utterance is a speech form able to form a word-meaning relation and serve as a means to convey meaning.[45]

The fact that speech is abiding is invoked by Kātyāyana and Patañjali to solve the problem of circularity in the use of terms such as *vr̥ddhi*. In the process of the generation of the speech form *mārṣṭi* (3sa pre) from the root *mr̥j* 'wipe', the *r̥* of the root is replaced by *ā* in accordance with 7.2.114 *mr̥jer vr̥ddhiḥ*, utilizing the term *vr̥ddhi*.[46] The sounds *ā*, *ai*, and *au* are termed *vr̥ddhi* in accordance with 1.1.1 *vr̥ddhir ād aic*. The problem is raised that 7.2.114 will be ineffective because the term *vr̥ddhi* can only apply to an *ā* that already exists at the time such sounds are termed *vr̥ddhi* in 1.1.1. Yet the *ā* in *mārṣṭi* doesn't exist when 1.1.1 applies; it is created by the application of the term *vr̥ddhi* which in turn denotes *ā* only by virtue of 1.1.1. Since 1.1.1 terms sounds *ā* that already exist *vr̥ddhi*, and 7.2.114 creates the *ā* in *mārṣṭi* by using the term *vr̥ddhi*, the rules are mutually dependent, the grammar involves circularity and fails.[47]

Under 1.1.45 *ig yaṇaḥ saṁprasāraṇam*, Patañjali discusses a similar situation in the case of the reference of the term *saṁprasāraṇa* to sounds *i*, *u*, *r̥*, and *l̥*. There a concept is introduced that is passed over in the discussion of the term *vr̥ddhi* under 1.1.1: the concept of a future term (*bhāvinī saṁjñā*). The term *saṁprasāraṇa* could be used to refer to the saṁprasāraṇa sounds *i*, *u*, *r̥*, and *l̥* that will be brought into existence. The analogy is made to a customer who approaches a weaver, hands him some thread and asks him to weave him a saree. Since the saree doesn't exist until after the threads are woven together, and one doesn't undertake the act of weaving on an already complete saree, the weaver understands that the customer uses the term *saree* as a future term: it refers to that which will be a saree once it has been woven.[48] A more familiar contemporary example might be the use of the term *cake* in the sentence, "Bake a cake." One bakes the ingredients that will be a cake once baked; one does not put a finished cake in the oven to bake.

The future-term explanation, however, is superseded, in the discussion of the term *saṁprasāraṇa* under 1.1.45,[49] in favor of another that is spelled out in greater detail in the discussion of the term *vr̥ddhi* under 1.1.1.[50] Under 1.1.1, Kātyāyana and Patañjali conclude that the procedure of the grammar succeeds because speech is abiding (*nitya*) (*siddhaṁ tu nityaśabdatvāt*).[51] The speech form *mārṣṭi* already exists, and the term *vr̥ddhi* refers to the *ā* in it that already exists. The objection is then raised that if speech forms are abiding and forms such as *mārṣṭi* already exist, there would be no purpose served by rule 7.2.114, which formally creates such speech forms, nor would there be any purpose served by generative grammar generally. This objection is met by reiterating that the rule prevents one from understanding that *mr̥j*, without *vr̥ddhi*, is correct everywhere; it instructs that the correct form is *mārj*, before affixes not

[45] For discussion of different views concerning the eternity of speech see Scharf 2006, esp. 141, 196-202, D'Sa 1980, Chakravarti 1933, and Gaurinath Sastri 1959.

[46] The *ā* is then followed immediately by *r* in accordance with 1.1.51 *uraṇ raparaḥ*.

[47] K1.40.18-21.

[48] K I.112.9-14.

[49] K I.112.14-17.

[50] K I.40.26-I.41.4.

[51] vt. 9. K I.40.26.

marked with *k* or *ṅ*. Since speech is abiding, the grammar serves the purpose, not of generating speech but of restricting usage to correct versus incorrect speech forms [52]

The apparent circularity in the case of semantic conditions that are defined in terms of speech forms can be solved in a similar way. The previous section pointed out that it is circular to define a semantic condition (e.g. *saptamyartha*) in terms of speech forms (e.g. *saptamī vibhakti*) that are generated by rules that include those semantic conditions. The circularity is avoided by understanding that the relationship between speech forms and their meaning is abiding. The rules do not actually generate the speech forms in certain meanings; they instruct one that it is correct to use certain speech forms in certain meanings. The linguistic description of the relation between the seventh vibhakti and its meanings is therefore timeless and legitimately referred to at any point in a derivation.

The terms *cake* and *saree* must be understood to refer to cakes and sarees generally, and to particular instances of cakes and sarees still to be produced, in order for ordinary affairs to be conducted successfully. These terms are so understood because the relation between speech forms and their meanings is virtually constant in the linguistic community. Similarly, for the successful procedure of the grammar, the terms *vṛddhi* and *saṁprasāraṇa* must be understood to refer to the sounds *ā*, *ai*, and *au;* and *i*, *u*, *ṛ*, and *ḷ*, respectively, even to particular instances of them that have not been generated by the formal procedure of the grammar. The terms are so understood because the grammar, although generative in form, is understood as instruction concerning the correct usage of a language that is virtually constant in the linguistic community. Likewise for the successful procedure of the grammar, the term *saptamī* in 1.1.19 must be understood to refer to a certain triplet of nominal terminations, even if the formal procedure of the grammar has not yet generated those nominal terminations, so that the term *saptamyartha* can be understood to refer to the semantic conditions for the occurrence of certain speech forms. The metalanguage used in the grammar must be available to the mechanics of the grammar at the time of procedural implementation of rules that use it, just as language is understood by people in the conduct of ordinary affairs.

Circularity is avoided in the use of speech forms to define semantic criteria that condition those speech forms because speech is abiding, and its relationship to meaning is established. Hence the meaning of the seventh vibhakti is known even before any particular derivational sequence is exhibited.

4.7 Kārakas

As early as 1964, R. Rocher (1964: 51) criticized the characterization of kārakas as syntactic categories, instead arguing that they are semantic. Calling them syntactico-semantic, Cardona (1976: 215-224) countered that it is suitable to consider kārakas as a level between the purely semantic level and the level at which nominal terminations are introduced (the abstract morphological level in Kiparsky 2002) because the rules that introduce kāraka terms include both semantic and co-occurrence conditions.

It is certainly the case that co-occurrence conditions enter into kāraka classification rules, and therefore that the kāraka classification is an intermediate stage of derivation

---

[52] vt. 10. *kimarthaṁ śāstram iti cen nivartakatvāt siddham.* K I.41.1.

between that of semantic conditions and that of the introduction of nominal terminations. It is possible that such an intermediate stage serves merely the purpose of procedural economy and does not imply that kāraka classification constitutes a level in any psychological or structural sense. Pāṇini may conceive of just two levels: semantic (*artha*) and phonetic (*śabda*). Kārakas are objects intended in certain relations; the level of intention is that of meaning, that is, the semantic level. One prominent seventeenth century Indian philosopher of language seems to favor the conception of kārakas as semantic categories. Kauṇḍabhaṭṭa in the *Subarthanirṇaya* of his *Vaiyākaraṇa-bhūṣaṇa-sāra* speaks of basic meanings for kārakas. He describes the rules that do not mention syntactic conditions as circumscribing general semantic domains for them. Yet the fact that Pāṇini formulated rules categorizing certain semantic items under certain syntactic conditions in exception to these domains may capture the conception, held by speakers of the language, of such categories as natural groups. Whether this sort of conceptualization comprises a level between the semantic and the morphological, or whether all conceptualization by virtue of being conceptual is semantic, is a moot point from the point of view of Pāṇinian procedure. In Pāṇinian procedure, kāraka classification does occupy an intermediate stage between purely semantic conditions and the introduction of speech forms. The intermediate stage is a way of achieving a complex mapping between meaning and speech.

Granted that procedurally kāraka rules intervene between semantics and phonetics and thereby serve as an interface between them. Yet the rules involve both semantics and co-occurrence conditions themselves and thereby the items classed by such rules are characterized by both semantic and phonetic parameters. From a psychological or structural perspective, therefore, the kāraka classification that results from kāraka rules constitutes a mixture of levels rather then an intermediate level.

4.8 *L*-Affixes

In their description of levels, Kiparsky and Staal place *l*-affixes at the same level as kārakas. Kiparsky (2002: 3) describes "Assignment of kārakas (Th-roles) and of abstract tense" as the function of the first set of rules mapping the semantic level to the morphosyntactic level. The treatment of *l*-affixes by Pāṇini, however, differs markedly from the treatment of kārakas. Kārakas are semantic objects classified by being designated by terms (*sañjñā*). Section 1.4 classifies semantic objects intended to be expressed by a speaker in relational categories by calling them by a kāraka term. Speech forms are subsequently introduced under the condition that an item designated by a kāraka term is to be denoted. *L*-affixes, in contrast, are introduced under semantic and syntactic conditions, just as other affixes are, and then are replaced by morphological elements; they serve therefore as abstract morphological elements themselves rather than as morphosyntactic representations.[53] Kiparsky differentiates abstract morphological representation from morphosyntactic representation. Therefore, if *l*-affixes belong to abstract morphological representation and kārakas to morphosyntactic representation, it is incorrect to assert that they occupy the same level in Pāṇinian grammar.

[53] Cardona (1997: 496) calls them "abstract affixes".

Part of the motivation for assigning *l*-affixes to the level of morphosyntactic representation and their replacements ***tip***, *tas*, *jhi*, etc. to the level of abstract morphological representation is to place the basic set of verbal terminations and the basic set of nominal terminations at the same level in the hierarchy and thereby to achieve parallelism between them. Just as the basic nominal terminations -***su***, -*au*, -***j****as*, etc. are distributed over semantic and syntactic conditions including kāraka and number, the basic verbal terminations -*ti****p***, -*tas*, -*jhi*, etc. are distributed over the same conditions kāraka and number, and similar conditions such as person (*puruṣa*). Kiparsky (2002: 3) calls the rules that achieve this distribution 'morphological spellout rules'. 3.4.78 *tiptasjhi...* introduces the basic set of verbal terminations just as 4.1.2 *svaujas...* introduces the basic set of nominal terminations. These sutras are read in conjunction with restrictive rules (*niyama*) that achieve the proper distribution over the conditions of number (1.4.21-22),[54] person (1.4.105-108),[55] and kāraka (pāda 2.3 for nominal terminations, and 1.3.13-93 for verbal terminations).

However, the parallelism is incomplete. The verbal terminations introduced by 3.4.78 are not distributed over the conditions of time and mood as the nominal terminations introduced by 4.1.2 are distributed over kārakas. On the contrary, it is rather the *l*-affixes introduced by 3.2.110 *luṅ*, 3.2.111 *anadyatane laṅ*, etc. that are distributed over time and mood. Moreover, *l*-affixes are distributed over certain kāraka conditions: 3.4.69 *laḥ karmaṇi ca bhāve cākarmakebhyaḥ* accounts for the distribution of *l*-affixes over karman or *bhāva* depending upon whether the root after which the *l*-affix occurs is transitive (*sakarmaka*) or intransitive (*akarmaka*). Verbal terminations, including the so called basic verbal terminations, are morphophonemic replacements of the *l*-affixes. On the grounds of the parallelism between *l*-affixes and basic nominal terminations, in addition to the fact that they, like the basic nominal terminations -***su***, -*au*, -***j****as*, etc. are initially introduced items rather than replacements, *l*-affixes, rather than the so called basic verbal terminations -*ti****p***, -*tas*, -*jhi*, etc., would properly be placed at the same level as basic nominal terminations in Kiparsky's fourfold hierarchy of levels.

Basic verbal terminations -*ti****p***, -*tas*, -*jhi*, etc. are therefore simply morphophonemic modifications of the *l* in *l*-affixes, just as, for example, the imperative terminations -*tu*, -*tām*, -*antu*, etc. are further morphophonemic modifications of the so-called basic verbal terminations -*ti****p***, -*tas*, -*jhi*, etc. and just as *ina*, *āt*, and *sya* (introduced after *a*-final stems by 7.1.12 *ṭāṅasiṅasām inātsyāḥ*) are morphophonemic modifications of the basic nominal terminations -***ṭ****ā*, -***ṅ****as****i***, and -***ṅ****as*.

4.9 Abstract Morphology Versus Phonology

The claim that the phonological output form resides on a different level from the abstract morphological representation is problematic. The abstract morphological representation often appears unchanged as the final phonological output, without having been subject to any additional rule. Many of the so-called basic verbal terminations,

[54] 1.4.21 *bahuṣu bahuvacanam*. 1.4.22 *dvyekayor dvivacanaikavacane*.

[55] 1.4.105 *yuṣmady upapade samānādhikaraṇe sthāniny api madhyamaḥ*. 1.4.106 *prahāse ca manyopapade manyater uttama ekavac ca*. 1.4.107 *asmady uttamaḥ*. 1.4.108 *śeṣe prathamaḥ*.

which Kiparsky placed on the level of abstract morphological representation (and which the last section argued are simply morphophonemic modifications of *l*-affixes) occur as the final phonological output of present active and imperfect middle and passive indicative verb forms in many contexts. The affix *-tas* for example, appears unchanged in *bhavatas* (3da pre of *bhū*) before *t* or *th*. In the example *devadatta odanaṁ pacati* discussed in section IVA above, (see Figure 4) the affix *-ti* in *pacati*, remains unchanged except for the dropping of the marker *p*. Basic nominal terminations often appear unchanged in final output form in many contexts. For example, the affix *-bhis* appears unchanged in *mālābhis* (f3s of *mālā*) before *t*, *th*. Can a string be on a different level from itself? In what sense of 'level' is this permissible? Note that Kiparsky (2002: 49) states that his scheme of levels "makes no distinction between 'phonology', 'morpho-phonology', and 'allomorphy'."

Now one can certainly argue that the choice of the particular abstract morphological representation is arbitrary and that it is just coincidental that in some cases the final output is identical to it. It is quite possible that one could select an abstract representation that never appears as phonological output. This is precisely what the previous section argued is the situation with the *l*-affixes. *L*, with various markers, is the abstract morphological representation of all verbal terminations. At least one stage of replacement for *l* always occurs to get the final output form of a verbal termination, whereas for nominals it is not necessarily the case that any additional stage occurs. Stages of replacement vary greatly in the production of speech forms; there is no clear association between those stages and any psychological or conceptual level. Three stages of replacement occur in the derivation of the form *bhavantu* (3pa ipv of *bhū*). (1) The *l* of *loṭ* is replaced by *jhi* by 3.4.78 *tiptasjhi...* (2) The *i* of *jhi* is replaced by *u* by 3.4.86 *er uḥ*. (3) The cover symbol *jh* is replaced by *ant* after *a*-final stems by 7.1.3 *jho 'ntaḥ*. Are we to posit three levels to correspond to these three stages of derivation? At least the use of the cover symbol *jh* achieves a valuable generalization in unifying the verbal terminations of the third person plural. Are we to posit an additional level at which such generalizations achieved by the use of cover symbols of this kind reside? The use of such cover symbols achieves an economy of rules in comparison to replacement of part or all of one basic termination that appears in phonetic output by sounds that appear in phonetic output in other contexts.[56] The use of *l*'s is essentially no different. If positing separate levels for cover symbols and their replacements is not procedurally justified, then what is the justification for positing separate levels for *l*-affixes and the basic verbal terminations that initially replace them? A twentieth century conception of syntax?

In distinction to potentially multiple stages of affixes and their replacements, it seems to me that just one level is involved once an affix has been introduced. The fact that Pāṇini uses the technique of replacement for the derivation of the final output form from an abstract morphological representation indicates that the replacement is considered to belong to the same level rather than to a different one; it belongs to the morphophonemic level as opposed to the semanticosyntactic level.

The semantic and syntactic levels are properly coalesced in a semantico-syntactic level and the abstract morphological and the morphophonemic levels are properly

---

[56] Cardona (1997: 330-332) discusses cover symbols and (490-492) demonstrates the economy of the inclusion of the cover symbol *jh* in the basic verbal terminations.

coalesced in a single morphophonemic level. While Pāṇini derives forms through numerous un-correlated stages of derivation, he makes a clear distinction between the level of meaning and the level of speech.

The concept of levels in Pāṇinian grammar, and the hierarchy of four levels proposed by Kiparsky and Staal, was inspired by divisions that evolved in modern linguistics. It is anachronistic to read them into the *Aṣṭādhyāyī*. Kiparsky himself (2002: 2) hedges his attribution of levels to Pāṇini calling them, "what we (from a somewhat anachronistic modern perspective) could see as different levels of representation." Pāṇini's grammar certainly worked with two levels: meaning and speech. Its derivational procedure certainly included more than two stages. However, it appears forced to press the derivational stages into a conceptual hierarchy of levels between the purely semantic and the purely phonetic, particularly into a four-level hierarchy corresponding to modern linguistic divisions.[57] Attempting to isolate syntax from semantics in the field of linguistics parallels the attempt to isolate relations from terms, and analytic statements from synthetic ones in the application of formal language models to natural language. Both are as indefensible as the isolation of forces from particles in classical physics has proven to be.[58]

In describing Pāṇinian procedure, one must be clear about when one is superimposing conceptions from contemporary linguistics on Pāṇini. Likewise, in modeling Pāṇinian procedure one must be clear about when one is introducing contemporary computational procedures foreign to Pāṇini. In the next section, I describe the organization of Pāṇinian grammar, purely from a Pāṇinian perspective rather than from the perspective of modern theoretical linguistics. In the remainder of this paper, I differentiate computational implementations of Pāṇinian grammar that model Pāṇinian procedure from applications of non-Pāṇinian generative computational techniques to Sanskrit.

5 Sketch of an Overview of Pāṇinian Architecture

The grammar is set up to derive correct speech forms from an open lexicon under certain conditions. The usual conditions are semantic, i.e. that certain meanings are to be denoted. Occasionally, conditions include pragmatics and literary context. In general, therefore, the grammar derives speech forms from meaning rather than vice versa. The grammar is not organized to determine the meaning of statements; it proceeds from the speakers point of view, not from the listeners point of view. It answers the question, "How do I say x?," not the question, "What does x mean?"

5.1 Introduction of Basic Elements on Semantic Conditions

In general Pāṇinian grammar introduces basic speech elements, or morphological elements, under semantic conditions. Basic speech elements include roots, nominal bases and affixes. Roots are introduced in two ways:

---

[57] Hyman (2003: 188-89) argues that Herodian's recognition of three types of linguistic errors--namely, barbarism, solecism, and acyrologia--corresponds to the threefold distinction of phonology, morphosyntax, and semantics.

[58] See W. V. O. Quine, "Two Dogmas of Empiricism," 2d, 1961 "The statement, rather than the term, came with Frege to be recognized as the unit accountable to an empiricist critique." http://www.ditext.com/quine/quine.html.

1. Elements listed in the *Dhātupāṭha* are termed roots (*dhātu*) by rule 1.3.1 *bhūvādayo dhātavaḥ.*
2. Derived elements terminating in any of a series of affixes introduced in rules 3.1.5-31 are termed roots by rule 3.1.32 *sanādyantā dhātavaḥ.*

Nominal bases are likewise introduced in two ways:

1. Any meaningful element other than a root (*dhātu*), affix (*pratyaya*), or an element that terminates in an affix, whether listed or not, is termed a nominal base (*prātipadika*) by 1.2.45 *arthavad adhātur apratyayaḥ prātipadikam.*
2. Derived elements, including both those terminating in affixes termed *kr̥t* or *taddhita* and compounds (*samāsa*), are termed nominal base by 1.2.46 *kr̥ttaddhitasamāsāś ca.*

Affixes are introduced by rules in adhyāyas 3-5 governed by the heading 3.1.1 *pratyayāḥ*. These include affixes in the list designated by 3.3.1 *uṇādayo bahulam.*

The basic speech elements of the grammar do not constitute a fully specified set of elements. First, lists are not specified as part of the ruleset; they are specified by commentators subsequently, which leaves open to doubt which items were intended to be included by the author of the rules himself. Second, the grammar includes recursive procedures. The derivates of certain procedures serve as conditions for other procedures which in turn serve as conditions for the first procedures. The derivational procedure permits the derivation of nominal bases from roots and other nominal bases, and the derivation of words from roots and nominal bases. The derivation procedure also permits the derivation of roots from roots, roots from nominal bases, roots from nominal words, and nominal bases from words.

Aside from lists being in doubt and the presence of recursive derivation of elements, the set of basic elements is an open set since what is classed as a nominal base includes any meaningful element outside of a specified set. 1.2.45 reads, "any meaningful element other than ... is a nominal base." Moreover, commentators call many of the lists of nominal bases merely paradigmatic (*ākr̥tigaṇa*) rather than complete. Finally, the fact that by 3.1.8-11 verbal roots are derived from an unspecified set of nominal words (*pada*), which are in turn derived from the open set of nominal bases, makes verbal roots an open set as well.

Now, nominal bases are explicitly stated to be meaningful, and affixes are introduced under semantic conditions. While no statement of the grammar introduces underived roots under semantic conditions, and the *Dhātupāṭha* list did not originally include semantic designations for them, they are assumed to be meaningful elements from the outset. Roots and nominal bases enter the grammar as speech forms after which affixes are provided under specified conditions, prevalently including semantic conditions. Roots and nominal bases enter the grammar by being refered to specifically, by being included in a list, or by the terms *dhātu* and *prātipadika* in the ablative (or occasionally the genitive) case (*dhātoḥ* 3.1.7, 3.1.91, *prādipadikāt* 4.1.1). The ablative (or genitive) in such rules indicates that after which affixes are provided.[59]

[59] Concerning the use of the ablative and genitive in such rules, see Scharf, in press.

5.2 Phonological Modification

Once basic elements have been introduced in chapters 3-5 of the *Aṣṭādhyāyī*, they are subject to morphophonemic operations taught in chapters 6-8. Introduced elements are subject to augmentation, and they (1.1.55 *anekāl śit sarvasya*) and their parts (1.1.52-54 *alo 'ntyasya*, etc.) are subject to replacement, and deletion. Some replacements have the status of their substituends (1.1.56 *sthānivad ādeśo 'nalvidhau*); others don't. Some types of affix-deletion (*luk*, *lup*, *ślu*) negate operations conditioned by the affix (1.1.63 *na lumatāṅgasya*); others (*lopa*) don't (1.1.62 *pratyayalope pratyayalakṣaṇam*).

Some of the operations that occur on introduced speech forms are cognizant of morpheme boundaries; others are not. Operations that are cognizant of morpheme boundaries take place on stems (*aṅga*) before affixes, on affixes after stems, or at word (*pada*), or sentence (*vākya*) boundaries, or on other entire meaningful units (*sarva*). Some take place only word-final (*padāntasya* 8.4.37, 8.4.59) or only not word-final (*apadāntasya* 8.3.24, 8.3.55). The rules in the section beginning with 6.4.1 *aṅgasya* and ending at the close of the seventh adhyāya recognize stem-affix boundaries. Rules in the sections beginning with 8.1.16 and 8.3.55 through the end of the third pāda of the eighth adhyāya are cognizant of pada boundaries. Rules 8.1.1-15 apply to entire meaningful units. Operations on introduced speech forms that are not cognizant of morpheme boundaries take place in continuous speech (*saṁhitā*) with no conditions other than phonetic context. Such rules are relatively few. Augmentation with *t* (*tuk*) and general vowel sandhi rules occur in the section following 6.1.72 *saṁhitāyām*, and general consonant sandhi rules occur at the end of the last pāda of the eighth adhyāya, beginning with 8.4.40 *stoḥ ścunā ścuḥ*. The sparsity of rules that are incognizant of morpheme boundaries testifies to the great extent to which syntacticosemantic conditions pervade morphophonemic operations.

6 Pāṇinian Procedure Versus non-Pāṇinian Generation

In order to illustrate the difference in approach required to create a computational model of Pāṇinian grammar as opposed to generating speech forms computationally without regard to Pāṇinian procedure, a few examples of how rules would be formulated under each approach are provided in the following sections. One example concerns the implementation of sandhi; two others concern nominal inflection and verbal inflection, respectively.

6.1 Sandhi

Without regard to Pāṇinian procedure, yet producing results consistent with Pāṇinian description, one could generate interword sandhi by constructing sandhi tables like the vowel-sandhi table shown in Table 8. Table 8 is Scharf's modification of Coulson's (1976) foldout vowel sandhi table. Rules would then be written simply to replace the left context, shown in the top row, and occasionally the right context, shown

in the right column, by the contents of the cell indexed by a cell in the top row and a cell in the right column. Items in bold show single replacements for both left and right contexts. Items in parenthesis show replacements just for the right context, and items in black italics show replacements for just the left context.

Table 8. Vowel Sandhi Table

ā̆	ī̆	ū̆	r̥	e	ai	o	au	
ā	*y*	*v*	*r*	e (’)	*ā*	o (’)	*āv*	| a
ā	*y*	*v*	*r*	*a*	*ā*	*a*	*āv*	| ā
e	***ī***	*v*	*r*	*a*	*ā*	*a*	*āv*	| ī̆
o	*y*	***ū***	*r*	*a*	*ā*	*a*	*āv*	| ū̆
ar	*y*	*v*	***r̥̄***	*a*	*ā*	*a*	*āv*	| r̥
ai	*y*	*v*	*r*	*a*	*ā*	*a*	*āv*	| e
ai	*y*	*v*	*r*	*a*	*ā*	*a*	*āv*	| ai
au	*y*	*v*	*r*	*a*	*ā*	*a*	*āv*	| o
au	*y*	*v*	*r*	*a*	*ā*	*a*	*āv*	| au

In contrast, to model Pāṇinian procedure requires creating data structures and a framework that allow one to approximate the statement of Pāṇinian rules in an executable language. Scharf (1992) wrote a Pascal program that executes sandhi between words and compound elements and presented the implementation at the 44th Annual Meeting of the Association for Asian Studies. In 2002, Scharf and Hyman designed a portable framework using modified regular expressions in an XML file to model Pāṇinian rules (Table 9). Each rule is written as one or more XML rule tags each of which contains several parameters: `source`, `target`, `lcontext`, `rcontext`, `optional`, and `c`. The optional parameters lcontext and rcontext specify the left and right contexts for the replacement of the source by the target. The optional parameter `optional` specifies that the current state is to be duplicated and subsequent parallel paths created, one in which the rule is implemented and the other in which it is not. The parameter `c` (for comment) contains the number of the Pāṇinian rule implemented by the rule tag. While most rules are implemented in a single rule tag, 6.1.101 requires five rule tags to implement. The implementation utilizes the Sanskrit Library Phonetic encoding scheme SLP1, in which Sanskrit sounds and common phonetic features such as tones and nasalization are each represented by a single character.[60]

Table 9. Pāṇinian Sandhi Rules

```
<!--acsandhi vowel sandhi-->
<rule source="([@(f)@(x)])([@(wb)])([@(f)@(x)])" tar-
  get="%(fxvarRa($1))$2%(fxvarRa($3))"  c="1.1.9 vt.
  fkAraxkArayoH savarRavidhiH"/>
<rule source="([@(a)])[@(wb)][@(a)]" tar-
  get="!(lengthen($1))" c="6.1.101"/>
<rule source="([@(i)])[@(wb)][@(i)]" tar-
  get="!(lengthen($1))" c="6.1.101"/>
<rule source="([@(u)])[@(wb)][@(u)]" tar-
```

[60] http://sanskritlibrary.org/encoding/SLP1.pdf

Table 9. (*continued*)

```
   get="!(lengthen($1))" c="6.1.101"/>
 <rule source="([@(f)])[@(wb)][@(f)]" tar-
   get="!(lengthen($1))" c="6.1.101"/>
 <rule source="([@(x)])[@(wb)][@(x)]" tar-
   get="!(lengthen($1))" c="6.1.101"/>
 <rule source="[@(a)][@(wb)]([@(ec)])" tar-
   get="!(vfdDiize($1))" c="6.1.88.  vfdDir eci"/>
 <rule source="[@(a)][@(wb)]([@(ik)])" tar-
   get="!(guRate($1))" c="6.1.87.  Ad guRaH"/>
 <rule source="([@(ik)])" target="%(semivowel($1))"
   rcontext="[@(wb)][@(ac)]" c="6.1.77.  iko  yaR aci"/>
 <rule source="a" target="'" lcontext="[@(eN)][@(wb)]"
   c="6.1.109.  eNaH padAntAd ati"/>
 <rule source="e" target="ay" rcontext="[@(wb)][@(ac)]"
   c="6.1.78.  eco 'yavAyAvaH"/>
 <rule source="o" target="av" rcontext="[@(wb)][@(ac)]"
   c="6.1.78"/>
 <rule source="E" target="Ay" rcontext="[@(wb)][@(ac)]"
   c="6.1.78"/>
 <rule source="O" target="Av" rcontext="[@(wb)][@(ac)]"
   c="6.1.78"/>
 <!--end acsandhi vowel sandhi-->
```

The rule syntax utilizes a number of macros that model Pāṇinian structures. Macros are used to model Pāṇinian sound classes: varṇa, varga, guṇa, vṛddhi, saṁprasāraṇa, etc.; to create pratyāhāras: *ak*, *aṇ*, *ik*, *yaṇ*, etc.; and to group sounds with common phonetic features: aspirated sounds, unaspirated sounds, voiced sounds, unvoiced sounds, etc. For example, the macros `@(f)` and `@(x)` in 1.1.9 vt. represent the varṇas *ṛ* and *ḷ* respectively. The macros `@(eN)` in 6.1.109 and `@(ac)` in 6.1.78 represents the pratyāhāras *eṅ* (monothongs) and *ac* (vowels), respectively. Mappings are used to map sets of sounds onto corresponding sounds, such as short vowels onto long, and unvoiced stops onto voiced stops. Functions, such as `lengthen`, `guRate`, and `vfdDiize`, utilize the mappings to facilitate implementation of common operations, namely, the replacement of a vowel by its corrresponding long vowel, guṇa vowel, or vṛddhi vowel, respectively. The functions `lengthen`, `vfdDiize`, and `guRate` are utilized in 6.1.101, 6.1.88, and 6.1.87, respectively. Their parameter ($1) is a regular expression reference to the contents of the source parameter that appears in parenthesis. In accordance with 1.1.51 *uraṇ raparaḥ*, the later two functions include the provision of *r* after replacement of the vowel *ṛ* by its corresponding guṇa vowel *a*.

Rules are not pre-selected by hand; rather they are triggered by data that meets the conditions for the application of the rule. Hyman wrote a Perl program that converts the XML file of regular expressions to Perl executable code. The model succeeds in encoding Pāṇinian rules in a manner that allows the rules that come into play to be tracked. Rule tracking has valuable research and pedagogical applications. Hyman (2007, 2008) describes the procedure by which the XML vocabulary to express Pāṇini's sandhi rules was developed and how a series of stages converts the rules not

only into executable Perl code, but also into a network, and a finite state transducer. The latter, being extremely fast, will permit realtime web use of the models. Huet (2005) describes his use of finite state transducers to analyze sandhi in continuous Sanskrit strings.

6.2 Nominal Inflection

Similar to the way in which an external sandhi table can be implemented without regard to Pāṇinian procedure, one could generate nominal declension without regard to Pāṇinian procedure, yet produce results consistent with Pāṇinian description. If one considers a nominal paradigm, such as that of the masculine noun *deva* in Table 10, it is evident that the element *dev* remains constant, while the remainder of the word varies in the paradigm. One can extract a set of endings proper to *a*-final masculine stems that consists of the varying segments, as shown in Table 11, and then draft a rule (**1**) by which one deletes the final *a* of the stem in any *a*-final masculine nominal and adds the *a*-final stem terminations to generate the stem's full declension.

Similarly, one can extract a set of endings proper to *jan*-final masculine stems by segmenting the string *jan*, which is constant in the paradigm of *rājan*, from the endings that vary, as shown in Table 12. One can then draft a rule (**2**) by which one deletes the final *an* of the stem of any *jan*-final masculine nominal and adds the *jan*-final stem terminations to generate the full declension of any *jan*-stem masculine nominal. A similar procedure allows one to analyze (Table 14) and draft a rule (**3**) for masculine stems ending in *C[vm]an*, where *C* is any consonant.

Table 10. Nominal Declension Table: *deva*

	singular	dual	plural
1	*devas*	*devau*	*devās*
v	*deva*	*devau*	*devās*
2	*devam*	*devau*	*devān*
3	*devena*	*devābhyām*	*devais*
4	*devāya*	*devābhyām*	*devebhyas*
5	*devāt*	*devābhyām*	*devebhyas*
6	*devasya*	*devayos*	*devānām*
7	*deve*	*devayos*	*deveṣu*

Table 11. Nominal Declension Table Analysis: *deva*

1	dev-as	dev-O	dev-As
v	dev-a	dev-O	dev-As
2	dev-am	dev-O	dev-An
3	dev-ena	dev-AByAm	dev-Es
4	dev-Aya	dev-AByAm	dev-eByas
5	dev-At	dev-AByAm	dev-eByas
6	dev-asya	dev-ayos	dev-AnAm
7	dev-e	dev-ayos	dev-ezu

Table 12. Nominal Declension Table Analysis: *rājan*

1	rAj-A	rAj-AnO	rAj-Anas
v	rAj-an	rAj-AnO	rAj-Anas
2	rAj-Anam	rAj-AnO	rAj-Yas
3	rAj-YA	rAj-aByAm	rAj-aBis
4	rAj-Ye	rAj-aByAm	rAj-aByas
5	rAj-Yas	rAj-aByAm	rAj-aByas
6	rAj-Yas	rAj-Yos	rAj-YAm
7	rAj-Yi/rAj-ani	rAj-Yos	rAj-asu

Table 13. Nominal Declension Table Analysis: *ātman*

1	Atm-A	Atm-AnO	Atm-Anas
v	Atm-an	Atm-AnO	Atm-Anas
2	Atm-Anam	Atm-AnO	Atm-anas
3	Atm-anA	Atm-aByAm	Atm-aBis
4	Atm-ane	Atm-aByAm	Atm-aByas
5	Atm-anas	Atm-aByAm	Atm-aByas
6	Atm-anas	Atm-anos	Atm-anAm
7	Atm-ani	Atm-anos	Atm-asu

declension of *a*-final masc. stem = d1 + masc. *a*-stem endings (1)
(*as*, *O*, *As*, ..., *e*, *ayos*, *ezu*)

declension of *jan*-final masc. stem = d2 + masc. *an*-stem endings (2)
(*A*, *AnO*, *Anas*, ..., *Yi*, *Yos*, *asu*)

declension of *C[vm]an*-final masc. stem = d2 + masc. *an*-stem endings (3)
(*A*, *AnO*, *Anas*, ..., *ani, anos, asu*)

This procedure was used by Scharf and Cheifetz in 1995 and more recently by Kulkarni (http://ltrc.iiit.net/~anusaaraka/) and by Huet (http://sanskrit.inria.fr/). While his procedure achieves a computational implementation of nominal declension, it ails to capture the generalization inherent in the Pāṇinian analysis that posits a basic et of nominal terminations for all nominal declension. Instead of one basic set of ominal terminations, one requires multiple sets of terminations each proper to a speific stem type (**1**-**3**). Huet's implementation achieves a partial generalization by imlementing sandhi rules such as retroflexion to the output rather than constructing eparate tables for *rāma* and *doṣa* in addition to a table for *deva*, for instance, to inlect *a*-stem masculine stems.

In contrast, the XML data structures utilized in the last section to model Pāṇinian andhi can be augmented to allow derivation of nominal stems as shown in Table 13. charf and Hyman implemented Pāṇinian nominal derivation by introducing an addiional parameter `morphid` in the XML rule tag and utilizing Scharf's (2002: 29-30) et of nominal inflection tags. In this implementation, rules are grouped in rulesets iven a `name` parameter that specifies three or more stages in derivation, including hanges to terminations, changes to stems, and sandhi. These rulesets are further rouped to apply to stems that match gender and phonological parameters.

Table 14. Pāṇinian Declension Rules

```
<ruleset name="a-stem_derivation">
<rule source="Bis" target="Es" lcontext="#" mor-
  phid="3p" c="7.1.9"/>
<rule source= A  target= ina  lcontext= #  morphid= 3s
  c="7.1.12"/>
<rule source="e" target="ya" lcontext="#" morphid="4s"
  c="7.1.13"/>
<rule source="as" target="At" lcontext="#" morphid="5s"
  c="7.1.12"/>
<rule source="as" target="sya" lcontext="#" mor-
  phid="6s" c="7.1.12"/>
<rule source="Am" target="n$1" lcon-
  text="[@(hrasva)IUA]#" morphid="6p" c="7.1.54"/>
<rule source="s" target="" lcontext="#" morphid="vs"
  c="6.1.69"/>
<rule source="as" target="I" lcon-
  text="(^praTama|^carama|taya|^alpa|^arDa|^katipaya)#"
  morphid="1p" optional="yes" c="7.1.17, 1.1.33"/>
</ruleset>

<ruleset name="a-stem_changes">
<rule source="a" target="e" rcontext="#[Bs]" mor-
  phid="p" c="7.3.103"/>
<rule source="a" target="A" rcontext="#[ynB]"
  c="7.3.102"/>
<rule source="a" target="e" rcontext="#os"
  c="7.3.104"/>
</ruleset>

<ruleset name="stem-ending_sandhi">
<rule source="#am$" target="#m" lcontext="[@(ak)]" mor-
  phid="[1v2]" c="6.1.107"/>
<rule source="#ad$" target="#d" lcontext="[@(ak)]" mor-
  phid="[1v2]" c="6.1.107, 7.1.25 Kasika karika"/>
<rule source="#" target="_#" lcontext="[@(a)]" rcon-
  text="[@(ic)]" morphid="[1v2]" c="6.1.104"/>
<rule source="#" target="_#" lcontext="[@(dIrGa)]"
  rcontext="[@(ic)]" morphid="[1v2]" c="6.1.105"/>
<rule source="#" target="_#" lcontext="[@(dIrGa)]"
  rcontext="as" morphid="[mf][1v]p" c="6.1.105"/>
<rule source="([@(ak)])#[@(ac)]" tar-
  get="%(lengthen($1))" morphid="[1v2]p" c="6.1.102"/>
<rule source="_" target="" morphid="[1v2]" c="6.1.104,
  6.1.105"/>
<rule source="s$" target="n" lcontext="[@(dIrGa)]" mor-
  phid="m2p" c="6.1.103"/>
<rule source="a#([@(guRa)])" target="$1" c="6.1.97 ato
  guRe"/>
</ruleset>
```

While the method adopted succeeds in producing nominal paradigms utilizing rules that capture what Pāṇinian rules do, it is limited in the extent to which it models Pāṇinian procedure. Pāṇinian rules form a single cascade and are selected solely by data that meets the conditions of the rule. The XML nominal declension procedure just described, on the other hand, selects stems for sets of rules selected in advance by hand for their known application to stems that meet the ruleset's selection criteria. Therefore, although the implementation utilizes a single set of basic terminations, and hence is an advance over the procedure that relies on multiple sets of nominal terminations, it is not a close model of Pāṇinian procedure.

6.3 Verbal Inflection

Verbal inflection can also be implemented by extracting multiple sets of terminations from various paradigms just as was done for non-Pāṇinian nominal declension. If one considers a verbal paradigm, such as the present active indicative of the root *bhū* in Table 15, one can segment the invariant string *bhav* from the variant strings *ati*, *atas*, etc. (Table 16). One can extract a set of endings proper to *a*-final present stems such as *bhava*, and draft a rule (**4**) for the derivation of any *a*-final present stem: delete the final *a* and add the *a*-stem terminations.

Table 15. Verbal Conjugation Table: *bhū*

	singular	dual	plural
3rd person	*bhavati*	*bhavatas*	*bhavanti*
2nd person	*bhavasi*	*bhavathas*	*bhavatha*
1st person	*bhavāmi*	*bhavāvas*	*bhavāmas*

Table 16. Verbal Conjugation Table Analysis: *bhū*

Bav-ati	Bav-atas	Bav-anti
Bav-asi	Bav-aTas	Bav-aTa
Bav-Ami	Bav-Avas	Bav-Amas

Table 17. Verbal Conjugation Table Analysis: *rudh*

ru-RadDi	ru-ndDas	ru-nDanti
ru-Ratsi	ru-ndDas	ru-ndDa
ru-RaDAmi	ru-nDAvas	ru-nDAmas

Table 18. Verbal Conjugation Table Analysis: *yuj*

yu-nakti	yu-Nktas	yu-Yjanti
yu-nakzi	yu-Nktas	yu-Nkta
yu-najmi	yu-Yjvas	yu-Yjmas

conjugation of *a*-final present stem = d1 + *a*-stem endings (4)
(*ati*, *atas*, *anti*; *asi*, *aTas*, *aTa*; *Ami*, *Avas*, *Amas*)

conjugation of class 7 *D*-final present stem with preceding *r* or *z* = d1 + [rz][@(vowel)]?*D*-stem class 7 endings (*RadDi*, *ndDas*, *nDanti*; *Ratsi*, *ndDas*, *ndDa*; *RaDmi*, *nDvas*, *nDmas*) (5)

conjugation of class 7 *j*-final present stem = d1 + *j*-stem class 7 endings (*nakti*, *Nktas*, *Yjanti*; *nakzi*, *Nktas*, *Nkta*; *najmi*, *Yjvas*, *Yjmas*) (6)

Similarly, consider the paradigm of the root *rudh* (Table 17). One can extract a set of active endings proper to class 7 present *dh*-final stems in which there is a preceding *r* or *ṣ*. One would have to segment only *ru* as the invariant string and infer endings *ṇaddhi*, *nddhas*, *ndhan*, etc. One could then draft the rule (**5**): delete the final sound of the root and add the endings for class 7 *dh*-final stems with preceding *r* or *ṣ*. Consider then the paradigm of *yuj* (Table 18). One would have to infer a separate set of endings *nakti*, *ṅktas*, *ñjanti*, etc. and formulate a separate rule (**6**) for *j*-final class 7 stems. Just as in the similar approach for nominal declension, this procedure fails to capture the generalization inherent in the Pāṇinian analysis. Pāṇini posits a single basic set of verbal terminations for all verbal declension (Table 19; 3.4.78). Instead of one basic set of verbal terminations, the non-Pāṇinian approach requires numerous sets of terminations each proper to one among many very specific stem types (**4-6**).

Table 19. Pāṇinian basic verbal terminations

	active			middle		
	singular	dual	plural	singular	dual	plural
3rd person	*ti**p***	*tas*	*jhi*	*ta*	*ātām*	*jha*
2nd person	*si**p***	*thas*	*tha*	*thās*	*āthām*	*dhvam*
1st person	*mi**p***	*vas*	*mas*	*i**ḍ***	*vahi*	*mahi**ṅ***

Scharf and Hyman successfully modeled Pāṇinian verbal conjugation by further enriching the XML structure utilized for nominal declension (Table 20). They added two parameters to the rule tag: `lexid`, and `root`. The former allows reference to the class of the root in the Pāṇinian *Dhātupāṭha*. The latter allows reference to the original form of the root even when the previous rules have modified the input string.

The parameter `morphid` utilizes Scharf's (2002: 30-31) verbal inflection tags. Rules are implemented in a single cascade that applies to all strings. Rule selection is

Table 20. Pāṇinian Verbal Conjugation

```
<grammar>
<affixes name="basic_verbal_active" c="3.4.78">
<suffix add="#ti;p" person="3" number="s"/>
<suffix add="#tas;" person="3" number="d"/>
<suffix add="#Ji;" person="3" number="p"/>

<suffix add="#si;p" person="2" number="s"/>
<suffix add="#Tas;" person="2" number="d"/>
<suffix add="#Ta;" person="2" number="p"/>

<suffix add="#mi;p" person="1" number="s"/>
<suffix add="#vas;" person="1" number="d"/>
<suffix add="#mas;" person="1" number="p"/>
</affixes>
```

solely on the basis of the data meeting the conditions of the rule, just like Pāṇinian rules. This implementation of verbal conjugation succeeds in achieving the Pāṇinian generalization of utilizing a single set of basic verbal terminations for all verbal stems. The implementation of verbal conjugation also surpasses the implementation of Pāṇinian nominal declension in that the verbal conjugation succeeds in adequately modeling Pāṇinian procedure.

The current implementation of verbal conjugation does have some limitations, however. It relies on intermediate stems extracted from Whitney's *Roots* for all but perfect (*liṭ*) and aorist optative (*āśīr-liṅ*) verb forms, implementing only stem-conjugation for the bulk of tenses and moods. For the latter two tenses and moods, however, the computational implementation approximates Pāṇinian procedure fairly closely for the derivation of final forms directly from Pāṇinian basic elements alone. As in the implementation of Pāṇinian nominal inflection, the implementation of Pāṇinian verbal inflection includes rule tracking so that a derivational history of the form can be provided.

We look forward to utilizing the enriched framework in a revised, more faithful model of Pāṇinian declension. We are currently enriching the XML tagset further to allow derivation of participle stems and hope to go on to implement derivational morphology generally.

6.4 Concluding Remarks

Modeling Pāṇinian derivational procedure not only provides useful research and pedagogical tools such as derivational rule histories for derived forms. More importantly, attempting to work out details of a computational implementation of Pāṇinian generative procedure illuminates the understanding of Pāṇini's method. Understanding Pāṇini's method better contributes to the improvement of linguistic methodology generally. Working out models of Sanskrit generative grammar also has direct benefits for Indological studies by bringing computational methods to assist philological work and other humanistic pursuits related to India, and by bringing Indology into the field of digital humanities.

References

Bharati, A., Chaitanya, V., Sangal, R.: Natural Language Processing: A Paninian Perspective. Prentice-Hall of India, New Delhi (1995)

Cardona, G.: Pāṇini: A Survey of Research. Mouton, The Hague (1976)

Chakravarti, P.C.: The Linguistic Speculations of the Hindus. University of Calcutta (1933)

Chomsky, N.: Syntactic Structures. Mouton, The Hague (1957)

Coulson, M.: Teach Yourself Sanskrit. Hodder and Stoughton, Kent (1976)

D'Sa, F.X.: Śabdaprāmāṇyam in Śabara and Kumārila: Towards a Study of the Mīmāṃsī Experience of Language. Publications of the De Nobili Research Library, no. 7, Vienna (1980)

Sastri, G.: The Philosophy of Word and Meaning: Some Indian approaches with special reference to the philosophy of Bhartṛhari. Calcutta Sanskrit College Research Series 5 (1959); Century Press, Calcutta (reprint, 1983)

Houben, J.: 'Meaning statements' in Panini's grammar: on the purpose and context of the Astadhyayi. Studien zur Indologie und Iranistik 22, 23–54 (1999) [2001]

Huet, G.: A functional toolkit for morphological and phonological processing, application to a Sanskrit tagger. Journal of Functional Programming 15(4), 573–614 (2005), http://yquem.inria.fr/~huet/PUBLIC/tagger.pdf

Hyman, M.: One word solecisms and the limits of syntax. In: Swiggers, P., Wouters, A. (eds.) Syntax in Antiquity. Orbis Supplementa, vol. 23, pp. 179–192. International Center of General Dialectology, Louvain (2003)

Hyman, M.: From Pāṇinian Sandhi to Finite State Calculus. In: Proceedings of the First International Sanskrit Computational Linguistics Symposium, Rocquencourt, France, October 29-31, pp. 13–21 (2007) (revised as Hyman 2008 in the present volume)

K The Vyākaraṇa-mahābhāṣya of Patañjali. Kielhorn, L.F (ed.) 3 vols. Third edition revised and furnished with additional readings references and select critical notes by K. V. Abhyankar. BORI, Pune (1962, 1965, 1972) (reprint, 1985)

Kiparsky, P.: On the Architecture of Pāṇini's Grammar. In: The conference on the Architecture of Grammar at the Central Institute of English and Foreign Languages in Hyderabad, January 15-17 (2002), http://www.stanford.edu/~kiparsky/ (Included in the present volume)

Kiparsky, P., Staal, J.F.: Syntactic and semantic relations in Pāṇini. Foundations of Language 5, 83–117 (1969)

Rocher, R.: 'Agent' et 'objet' chez Pāṇini. Journal of the American Oriental Society 84, 44–54 (1964)

Rocher, R.: *Bhāva* 'état' et *kriyā* 'action' chez Pāṇini. In: Lebrun, Y. (ed.) Recherches linguistiques en Belgique, pp. 113–120. Universa, Wetteren (1966)

Scharf, P.: Sanskrit Sandhi from Pāṇini to Pascal. In: The 44th Annual Meeting of the Association for Asian Studies, Washington, D.C, April 2-5 (1992)

Scharf, P.: Early Indian Grammarians on a speaker's intention. Journal of the American Oriental Society 115(1), 66–76 (1995)

Scharf, P.: The Denotation of Generic Terms in Ancient Indian Philosophy: Grammar, Nyāya, and Mimāṁsā. Transactions of the American Philosophical Society, vol. 86, part 3. American Philosophical Society, Philadelphia (1996)

Scharf, P.: Rāmopākhyāna–the Story of Rāma in the Mahābhārata: An Independent-study Reader in Sanskrit. RoutledgeCurzon, London (2002)

Scharf, P.: Pāṇinian accounts of the Vedic subjunctive: leṭ kṛṇvaíte. Indo-Iranian Journal 48(1), 71–96 (2005); paper presented at The 214th Meeting of the American Oriental Society, San Diego, California, March 12-15 (2004), http://www.language.brown.edu/Sanskrit/ScharfSubjunctive.pdf

Scharf, P.: Pāṇinian accounts of the Vedic subjunctive: leṭ kṛṇvaíte. Indo-Iranian Journal 51(1), 1–21 (2008) (Corrected version of Scharf 2005)

Scharf, P.: The Natural-language Foundation of Metalinguistic Case-use in the Aṣṭādhyāyī and Nirukta. In: Cardona, G., Deshpande, M. (eds.) Proceedings of the 12th World Sanskrit Conference, Helsinki, Finland, July 13-18, 2003. Motilal Banarsidass, Delhi (in press)

Scharf, P.: Pāṇinian accounts of the class eight presents. Journal of the American Oriental Society (forthcoming); paper presented at the 13th World Sanskrit Conference, July 10-14, Edinburgh (2006)

Simulating the Pāṇinian System of Sanskrit Grammar

Anand Mishra

Ruprecht Karls University, Heidelberg, Germany
amishra@ix.urz.uni-heidelberg.de
http://sanskrit.sai.uni-heidelberg.de

Abstract. We propose a model for computer representation of the Pāṇinian system of sanskrit grammar. Based on this model, we render the grammatical data and simulate the rules of Aṣṭādhyāyī on computer. We then employ these rules for generation of morpho-syntactical components of the language. These generated components we store in a p-subsequential transducer. This we use to develop a lexicon on Pāṇinian principles. We report briefly its implementation.

Keywords: Sanskrit, Grammar, Panini, Ashtadhyayi, Computational linguistics.

1 A Representation of Aṣṭādhyāyī

The general grammatical process of Aṣṭādhyāyī [3] can be viewed as consisting of the following three basic steps:

1. Prescription of the fundamental components which constitute the language.
2. Characterization of these fundamental components by assigning them a number of attributes.
3. Specification of grammatical operations based on the fundamental components and their attributes.

1.1 Fundamental Components

In his grammar Pāṇini furnishes a number of elements (phonemes, morphemes and lexemes) which constitute the building blocks of the language. We assign each of them a unique key in our database. Thus the phoneme /*a*/ has the key `a_0`, the *kṛt* suffix /*a*/ has the key `a_3` and the *taddhita* suffix /*a*/ is represented by the key `a_4`. Given such a collection of unique keys, we define the set of fundamental components as follows:

Definition 1. *The collection of unique keys corresponding to the basic constituents of the language, we define as the set $\mathcal{F}$ of fundamental components.*

G. Huet, A. Kulkarni, and P. Scharf (Eds.): Sanskrit CL 2007/2008, LNAI 5402, pp. 127–138, 2009.

Further, we decompose this set $\mathcal{F}$ into two disjoint sets $\mathcal{P}$ and $\mathcal{M}$, where $\mathcal{P}$ is the set of keys corresponding to the phonemes and $\mathcal{M}$ containing the keys of the rest of the constituting elements (morphemes or lexemes). Thus,

$$\mathcal{P} = \{\texttt{a_0}, \texttt{i_0}, \texttt{u_0}, \ldots\} \tag{1}$$

$$\mathcal{M} = \{\texttt{bhU_a}, \texttt{tip_0}, \texttt{laT_0}, \ldots\} \tag{2}$$

and

$$\mathcal{F} = \mathcal{P} \cup \mathcal{M} \tag{3}$$

$$\mathcal{P} \cap \mathcal{M} = \phi \tag{4}$$

Remark 1. The set of keys corresponding to the phonemes $\mathcal{P}$, and the set of the keys belonging to the rest of the constituting elements (morphemes or lexemes) $\mathcal{M}$ are mutually disjoint.

1.2 Attributes

The fundamental units of the language are given an identity by assigning a number of attributes to them. We include the various technical terms introduced in the grammar under this category, as also the *it* -markers and sigla or *pratyāhāra*s. For example, the attributes *hrasva*, *guṇa* and *a*C characterize the element `a_0` as short vowel /*a*/ and attributes like *pratyaya*, *prathama* and *ekavacana* tell that *ti*P (= `tip_0`) is a third-person singular suffix. Again, each attribute is assigned a unique key in our database.

Definition 2. *The collection of unique keys corresponding to the terms, which characterize a fundamental component, we define as the set* $\mathcal{A}$ *of attributes.*

Corresponding to the sets $\mathcal{P}$ and $\mathcal{M}$ we can decompose the set $\mathcal{A}$ into two disjoint sets $\mathcal{A}_\pi$ and $\mathcal{A}_\mu$, $\mathcal{A}_\pi$ being the set of unique keys of the attributes to the elements of $\mathcal{P}$ and $\mathcal{A}_\mu$ to elements of $\mathcal{M}$.

$$\mathcal{A}_\pi = \{\texttt{hrasva_0}, \texttt{udAtta_0}, \texttt{it_0}, \ldots\} \tag{5}$$

$$\mathcal{A}_\mu = \{\texttt{dhAtu_0}, \texttt{pratyaya_0}, \texttt{zit_9}, \ldots\} \tag{6}$$

$$\mathcal{A} = \mathcal{A}_\pi \cup \mathcal{A}_\mu \tag{7}$$

Remark 2. Any two of the four sets $\mathcal{P}, \mathcal{M}, \mathcal{A}_\pi, \mathcal{A}_\mu$ are mutually disjoint.

2 Basic Data Structures

Given the set of fundamental components ($\mathcal{F} = \mathcal{P} \cup \mathcal{M}$) and the set of attributes ($\mathcal{A} = \mathcal{A}_\pi \cup \mathcal{A}_\mu$), we now define our data structure for representing the Pāṇinian process.

2.1 Sound Set ψ

Definition 3. *A sound set ψ is a collection of elements from sets $\mathcal{P}, \mathcal{M}$ and $\mathcal{A}$ having exactly one element from the set $\mathcal{P}$.*

$$\psi = \{\pi_p, \mu_i, \alpha_j | \pi_p \in \mathcal{P}, \mu_i \in \mathcal{M}, \alpha_j \in \mathcal{A}, i, j \geq 0\} \tag{8}$$

This is an abstract data structure. Although it corresponds to a phoneme or one sound unit, it represents more than just a phoneme (see Sec. 2.2).

2.2 Language Component $\boldsymbol{\lambda}$

Definition 4. *A language component λ is an ordered collection of at least one or more sound sets.*

$$\lambda = [\psi_0, \psi_1, \psi_2, \ldots \psi_n] \textit{ such that } \|\lambda\| > 0 \tag{9}$$

Note 1. We use square brackets [ ] to represent an ordered collection and curly brackets { } for an unordered collection.

Language expressions at every level (phonemes, morphemes, lexemes, words, sentences) can now be represented as a language component.

Example 1. We represent the verbal root *bhū* as a language component λ.

$$\begin{aligned} \lambda &= [\psi_1, \psi_2] \text{ where} \\ \psi_1 &= \{\textit{bh}, \textit{bhū}, \textit{dhātu}, \ldots\} \\ \psi_2 &= \{\textit{u}, \textit{bhū}, \textit{dhātu}, \textit{udātta}, \textit{dīrgha}, a\textsc{c}, \ldots\} \end{aligned}$$

Corresponding to the two phonemes *bh* and *ū* in *bhū*, we have an ordered collection of two sound sets ψ_1 and ψ_2. Consider the first one ψ_1: Its first element *bh* is from the phoneme set $\mathcal{P}$.[1] The second element *bhū* tells that the phoneme of this sound set is a part of the fundamental unit *bhū*. The third element stores the attribute *dhātu* (verbal root) to this sound set. Similarly, the second sound set ψ_2 has phoneme attributes which tell it to be an *udātta* (high pitched) *dīrgha* (long) *a*C (vowel).

Example 2. The language component corresponding to the morpheme *l*AṬ is represented as:

$$\begin{aligned} \lambda &= [\psi] \text{ where} \\ \psi &= \{\textit{l}, l\text{AṬ}, \textit{pratyaya}, \textit{ait}, \textit{ṭit}, \ldots\} \end{aligned}$$

---

[1] Actually, it is the unique key `bh_0` corresponding to the phoneme *bh* which is stored in the sound set, but for the sake of simplicity and readability we write the conventional letters for phonemes or morphemes.

Remark 3. Attribute *ṭit* says that it has *ṭ* as *it* - marker.

Example 3. The morphemes *bhū* followed by *l*AṬ can now be represented together by the language component

$$\lambda = [\psi_1, \psi_2, \psi_3] \text{ where}$$
$$\psi_1 = \{bh, bh\bar{u}, dh\bar{a}tu, \ldots\}$$
$$\psi_2 = \{u, bh\bar{u}, dh\bar{a}tu, ud\bar{a}tta, d\bar{\imath}rgha, a\textsc{c}, \ldots\}$$
$$\psi_3 = \{l, l\textsc{aṭ}, pratyaya, ait, ṭit, \ldots\}$$

Different linguistic units can now be identified by carrying out intersection with the appropriate subsets of $\mathcal{P}, \mathcal{M}$ and $\mathcal{A}$. For example to get the verbal root or *dhātu* in λ we take the intersection of an identity set $\iota = \{dh\bar{a}tu\}$ with each of ψ_i's in λ and store the index i when the intersection-set is not empty. In this case we get the index list [1,2]. The list of ψ_i's corresponding to these indices then gives the searched morpheme. Thus, the verbal root is given by the language component $[\psi_1, \psi_2]$.

2.3 Process Strip σ

Definition 5. *A process strip σ is an ordered collection of pairs, where the first element of the pair is the number of a particular grammatical rule (e.g. $rule_p$) and the second element is a language component λ.*

$$\sigma = [(rule_p, \lambda_p), (rule_q, \lambda_q), \ldots] \tag{10}$$

The rule number corresponds to the Aṣṭādhyāyī order and binds the process strip with a function implementing that rule in the actual program. Thus, the process strip simulates the Pāṇinian process by storing in the language component λ_p the effect of applying the rule $rule_p$.

3 Basic Operations

Having defined our data-structure in Sec. 2, we now introduce the basic operations on them.

3.1 Attribute Addition

Let $\alpha \subset \mathcal{A} \cup \mathcal{M}$ and ψ be a sound set. Then attribute addition is defined as

$$h_{a\psi}(\psi, \alpha) = \psi \cup \alpha \tag{11}$$

Remark 4. This operation can be applied to a number of sound sets given by indices $[i, i+1, \ldots, j]$ in a given language component λ

$$h_{a\lambda}(\lambda, \alpha, [i, \ldots, j]) = [\psi_1, \ldots, \psi_i \cup \alpha, \ldots, \psi_j \cup \alpha, \ldots, \psi_n] \tag{12}$$

Example 4. Let us consider the language component corresponding to the morpheme Śaᴘ. It is represented by

$$\lambda = [\psi] \text{ where}$$
$$\psi = \{a, \text{Śa}\textsc{p}, \textit{pratyaya}, \textit{śit}, \textit{pit}, \ldots\}$$

Rule tiṅ śit sārvadhātukam (3.4.113)[2] says that affixes in the siglum *ti*Ṅ and those having *ś* as *it* marker are assigned the attribute *sārvadhātuka.*

We implement this rule by checking if there are sound sets with attributes *pratyaya* together with *ti*Ṅ or *śit* and adding the attribute *sārvadhātuka* if the condition is fulfilled. In this case, we get:

$$\lambda = [\psi] \text{ where}$$
$$\psi = \{a, \text{Śa}\textsc{p}, \textit{pratyaya}, \textit{śit}, \textit{pit}, \textit{sārvadhātuka}, \ldots\}$$

3.2 Augmentation

Let

$$\lambda = [\psi_1, \ldots, \psi_i, \psi_{i+1}, \ldots, \psi_n]$$
$$\lambda_k = [\psi_{1k}, \psi_{2k}, \psi_{3k}, \ldots, \psi_{mk}]$$

and i be an integer index such that $i \leq \|\lambda\|$, then augmentation of λ by λ_k at index i is defined as

$$h_g(\lambda, \lambda_k, i) = [\psi_1, \ldots, \psi_i, \psi_{1k}, \psi_{2k}, \psi_{3k}, \ldots, \psi_{mk}, \psi_{i+1}, \ldots, \psi_n] \tag{13}$$

Example 5. Consider the language component λ corresponding to the verbal root *bhū.*

$$\lambda = [\psi_1, \psi_2] \text{ where}$$
$$\psi_1 = \{\textit{bh}, \textit{bhū}, \textit{dhātu}, \ldots\}$$
$$\psi_2 = \{\textit{u}, \textit{bhū}, \textit{dhātu}, \textit{udātta}, \textit{dīrgha}, a\textsc{c}, \ldots\}$$

Rule vartamāne laṭ (3.2.123) says that the morpheme *l*AṬ is added after a *dhātu* if the present action is to be expressed.

To implement this rule, we first look for the indices of sound sets which have the attribute *dhātu* and then append the sound set corresponding to *l*AṬ after the last index. We get,

$$\lambda = [\psi_1, \psi_2, \psi_3] \text{ where}$$
$$\psi_1 = \{\textit{bh}, \textit{bhū}, \textit{dhātu}, \ldots\}$$
$$\psi_2 = \{\textit{u}, \textit{bhū}, \textit{dhātu}, \textit{udātta}, \textit{dīrgha}, a\textsc{c}, \ldots\}$$
$$\psi_3 = \{\textit{l}, l\text{AṬ}, \textit{pratyaya}, \textit{ait}, \textit{ṭit}, \ldots\}$$

[2] Numbers like (3.4.113) correspond to the rule number in Katre's edition of Aṣṭādhyāyī [3].

3.3 Substitution

We define substitution in terms of the above two operations.

Let $[i, i+1, i+2, \ldots, j]$ be the indices of sound sets to be replaced in the language component $\lambda = [\psi_1, \ldots, \psi_i, \psi_{i+1}, \ldots, \psi_n]$.

Let $\lambda_k = [\psi_{1k}, \psi_{2k}, \psi_{3k}, \ldots, \psi_{mk}]$ be the replacement, then the substitution is defined as

$$h_s(\lambda, \lambda_k, [i, \ldots, j]) = h_g(h_{a\lambda}(\lambda, \{\delta\}, [i, \ldots, j]), \lambda_k, j) \tag{14}$$

where $\delta \in \mathcal{A}$ is the *attribute* which says that this sound set is no more active and has been replaced by some other sound set.

Example 6. Consider the language component corresponding to the verbal root ṇīÑ

$$\begin{aligned} \lambda &= [\psi_1, \psi_2] \text{ where} \\ \psi_1 &= \{\text{ṇ}, \text{ṇīÑ}, \textit{dhātu}, \textit{ñit}, \ldots\} \\ \psi_2 &= \{\textit{i}, \text{ṇīÑ}, \textit{dhātu}, \textit{ñit}, \textit{dīrgha}, a\text{C}, \ldots\} \end{aligned}$$

Rule ṇaḥ naḥ (6.1.065) says that the initial retroflex *ṇ* of a *dhātu* is replaced by dental *n*.

To implement this rule we first search the sound sets corresponding to *dhātu*, check whether the first one has a retroflex *ṇ* and if the conditions are fulfilled, add the attribute δ in that sound set and append the sound set corresponding to *n* after it. Further we transfer all attributes (except the phoneme attributes) from the *ṇ* - sound set to *n* - sound set for *sthānivadbhāva*. We get,

$$\begin{aligned} \lambda &= [\psi_1, \psi_2, \psi_3] \text{ where} \\ \psi_1 &= \{\text{ṇ}, \text{ṇīÑ}, \textit{dhātu}, \textit{ñit}, \delta, \ldots\} \\ \psi_2 &= \{\textit{n}, \text{ṇīÑ}, \textit{dhātu}, \textit{ñit}, \ldots\} \\ \psi_3 &= \{\textit{i}, \text{ṇīÑ}, \textit{dhātu}, \textit{ñit}, \textit{dīrgha}, a\text{C}, \ldots\} \end{aligned}$$

4 Grammatical Process

Having defined our basic data structure in Sec. 2 and primary operations on them in Sec. 3, we now represent a rule of grammar based on the above mentioned definitions.

4.1 Representing a Rule of Grammar

We represent a rule of grammar through a function f_q, which takes a process strip σ_p and adds a new pair $(rule_q, \lambda_q)$ to it where $rule_q$ is the number of the present rule and λ_q is the new modified language component after application

of one or more of the three operations defined above on the input language component λ_p.

$$\begin{aligned} f_q(\sigma_p) &= \sigma_q \text{ where} \\ \sigma_p &= [\ldots, (rule_p, \lambda_p)] \\ \sigma_q &= [\ldots, (rule_p, \lambda_p), (rule_q, \lambda_q)] \\ \lambda_q &= h_a, h_g, h_s(\lambda_p, \ldots) \end{aligned}$$

4.2 Structure of a Rule

The general structure of a rule is as follows:

`Function` f_q `with input strip:` $\sigma_p = [\ldots, (rule_p, \lambda_p)]$

`check applicability conditions`
if `conditions` *not* `fulfilled` *then*
    return `unextended` σ_p
else
   `create new modified` λ_q
   return `extended` σ_q

Thus, given a particular state (represented by σ_p) in the process of generation, the system provides for checking the applicability of a rule f_q, and if the conditions are fulfilled, the rule is applied and the changed language component together with the rule number is stored in the modified state (represented by σ_q).

As the rule numbers are also stored, we can implement the rules of *tripādī* and make their effects invisible for subsequent applications. The order in which rules are applied is provided manually through templates.

5 An Example

We take a verbal root *bhū* and generate the final word *bhavati* meaning "he/she/it becomes". We initialize the process strip σ_0 by loading the language component corresponding to the verbal root and adding $a00000$ as the rule number.

$$\begin{aligned} \sigma_0 &= [(a00000, \lambda_0)] \text{ where} \\ \lambda_0 &= [\psi_{0a}, \psi_{0b}] \text{ with} \\ \psi_{0a} &= \{bh, bh\bar{u}, dh\bar{a}tu, \ldots\} \\ \psi_{0b} &= \{u, bh\bar{u}, dh\bar{a}tu, d\bar{\imath}rgha, ud\bar{a}tta, \ldots\} \end{aligned}$$

Rule vartamāne laṭ (3.2.123) says that the morpheme *l*aṭ is added after a *dhātu* if the present action is to be expressed.

The application now involves following steps: Look in the last language component λ of the process-strip σ. If there are sound sets ψ with the identity set

$\iota = \{$ *dhātu* $\}$ in it, get their indices in index list. This returns the index list [1,2]. If index list is non empty then augment the language component λ_0 by attaching the language component corresponding to the morpheme *l*AṬ. This is attached in this case at index 2 as the new morpheme comes after *dhātu*. Extend the process strip σ_0 accordingly. Thus, we have

$$f_{a32123}(\sigma_0) = \sigma_1 \tag{15}$$

$$\begin{aligned} \sigma_1 &= [(a00000, \lambda_0), (a32123, \lambda_1)] \text{ where} \\ \lambda_1 &= [\psi_{0a}, \psi_{0b}, \psi_{1a}] \text{ with} \\ \psi_{0a} &= \{\textit{bh}, \textit{bhū}, \textit{dhātu}, \ldots\} \\ \psi_{0b} &= \{\textit{u}, \textit{bhū}, \textit{dhātu}, \textit{dīrgha}, \textit{udātta}, \ldots\} \\ \psi_{1a} &= \{\textit{l}, \textit{l}\text{AṬ}, \textit{pratyaya}, \textit{ait}, \textit{ṭit}, \ldots\} \end{aligned}$$

RULE tip tas jhi sip thas tha mip vas mas ta ātām jha thās āthām dhvam iṭ vahi mahiṅ (3.4.078) provides for substitution of *l*AṬ.

We take the first suffix *ti*P for replacement. The sound sets to be replaced are determined by taking intersection with the set {*l*AṬ, *l*IṬ, *l*OṬ, ...} which has the morphemes having cover term *l*. In this case it is at the index 3. We replace this sound set with *ti*P i.e. add the attribute δ to the sound set at index 3 and augment the language component at this index. We get,

$$f_{a34078}(\sigma_1) = \sigma_2 \tag{16}$$

$$\begin{aligned} \sigma_2 &= [\ldots, (a32123, \lambda_1), (a34078, \lambda_2)] \\ \lambda_2 &= [\psi_{0a}, \psi_{0b}, \psi_{1a}, \psi_{2a}, \psi_{2b}] \\ \psi_{0a} &= \{\textit{bh}, \textit{bhū}, \textit{dhātu}, \ldots\} \\ \psi_{0b} &= \{\textit{u}, \textit{bhū}, \textit{dhātu}, \textit{dīrgha}, \textit{udātta}, \ldots\} \\ \psi_{1a} &= \{\textit{l}, \textit{l}\text{AṬ}, \textit{pratyaya}, \textit{ait}, \textit{ṭit}, \delta, \ldots\} \\ \psi_{2a} &= \{\textit{t}, \textit{ti}\text{P}, \textit{pratyaya}, \textit{sārvadhātuka}, \textit{pit}, \ldots\} \\ \psi_{2b} &= \{\textit{i}, \textit{ti}\text{P}, \textit{pratyaya}, \textit{sārvadhātuka}, \textit{pit}, \ldots\} \end{aligned}$$

RULE kartari śap (3.1.068) says that the morpheme Ś*a*P is added after *dhātu* but before *sārvadhātuka* suffix and denotes agent.

Check if sound set with *sārvadhātuka* follows one with *dhātu*. If yes then augment the language component for Ś*a*P after *dhātu*.

$$f_{a31068}(\sigma_2) = \sigma_3 \tag{17}$$

$$\begin{aligned} \sigma_3 &= [\ldots, (a34078, \lambda_2), (a31068, \lambda_3)] \\ \lambda_3 &= [\psi_{0a}, \psi_{0b}, \psi_{3a}, \psi_{1a}, \psi_{2a}, \psi_{2b}] \\ \psi_{0a} &= \{\textit{bh}, \textit{bhū}, \textit{dhātu}, \ldots\} \end{aligned}$$

$$\begin{aligned}
\psi_{0b} &= \{u, bh\bar{u}, dh\bar{a}tu, d\bar{\imath}rgha, ud\bar{a}tta, \ldots\}\\
\psi_{3a} &= \{a, \acute{s}a\textsc{p}, pratyaya, hrasva, \acute{s}it, pit, \ldots\}\\
\psi_{1a} &= \{l, l\textsc{aṭ}, pratyaya, ait, \d{t}it, \delta, \ldots\}\\
\psi_{2a} &= \{t, ti\textsc{p}, pratyaya, s\bar{a}rvadh\bar{a}tuka, pit, \ldots\}\\
\psi_{2b} &= \{i, ti\textsc{p}, pratyaya, s\bar{a}rvadh\bar{a}tuka, pit, \ldots\}
\end{aligned}$$

Rule yasmāt pratyaya vidhis tad ādi pratyaye aṅgam (1.4.013) makes the part before the suffix ŚaP an *aṅga* with respect to it.

$$f_{a14013}(\sigma_3) = \sigma_4 \tag{18}$$

$$\begin{aligned}
\sigma_4 &= [\ldots, (a31068, \lambda_3), (a14013, \lambda_4)]\\
\lambda_4 &= [\psi_{0a}, \psi_{0b}, \psi_{3a}, \psi_{1a}, \psi_{2a}, \psi_{2b}]\\
\psi_{0a} &= \{bh, bh\bar{u}, dh\bar{a}tu, a\dot{n}ga, \ldots\}\\
\psi_{0b} &= \{u, bh\bar{u}, dh\bar{a}tu, a\dot{n}ga, d\bar{\imath}rgha, ud\bar{a}tta, \ldots\}\\
\psi_{3a} &= \{a, \acute{s}a\textsc{p}, pratyaya, hrasva, \acute{s}it, pit, \ldots\}\\
\psi_{1a} &= \{l, l\textsc{aṭ}, pratyaya, ait, \d{t}it, \delta, \ldots\}\\
\psi_{2a} &= \{t, ti\textsc{p}, pratyaya, s\bar{a}rvadh\bar{a}tuka, pit, \ldots\}\\
\psi_{2b} &= \{i, ti\textsc{p}, pratyaya, s\bar{a}rvadh\bar{a}tuka, pit, \ldots\}
\end{aligned}$$

Rule sārvadhātuka ārdhadhātukayoḥ (7.3.084) says that before *sārvadhātuka* or *ārdhadhātuka* replace the iK vowels by *guṇa* vowels.

As ŚaP is *sārvadhātuka*, we get

$$f_{a73084}(\sigma_4) = \sigma_5 \tag{19}$$

$$\begin{aligned}
\sigma_5 &= [\ldots, (a14013, \lambda_4), (a73084, \lambda_5)]\\
\lambda_5 &= [\psi_{0a}, \psi_{0b}, \psi_{5a}, \psi_{3a}, \psi_{1a}, \psi_{2a}, \psi_{2b}]\\
\psi_{0a} &= \{bh, bh\bar{u}, dh\bar{a}tu, a\dot{n}ga, \ldots\}\\
\psi_{0b} &= \{u, bh\bar{u}, dh\bar{a}tu, a\dot{n}ga, d\bar{\imath}rgha, ud\bar{a}tta, \delta, \ldots\}\\
\psi_{5a} &= \{o, bh\bar{u}, dh\bar{a}tu, a\dot{n}ga, \ldots\}\\
\psi_{3a} &= \{a, \acute{s}a\textsc{p}, pratyaya, hrasva, \acute{s}it, pit, \ldots\}\\
\psi_{1a} &= \{l, l\textsc{aṭ}, pratyaya, ait, \d{t}it, \delta, \ldots\}\\
\psi_{2a} &= \{t, ti\textsc{p}, pratyaya, s\bar{a}rvadh\bar{a}tuka, pit, \ldots\}\\
\psi_{2b} &= \{i, ti\textsc{p}, pratyaya, s\bar{a}rvadh\bar{a}tuka, pit, \ldots\}
\end{aligned}$$

Rule ec aḥ ay av āy āv aḥ (6.1.078) says that before aC (vowel) *e*, *o*, *ai*, *au* are respectively replaced by *ay*, *av*, *āy*, *āv*.

$$f_{a61078}(\sigma_5) = \sigma_6 \tag{20}$$

$$
\begin{aligned}
\sigma_6 &= [\ldots, (a73084, \lambda_5), (a61078, \lambda_6)] \\
\lambda_6 &= [\psi_{0a}, \psi_{0b}, \psi_{5a}, \psi_{6a}, \psi_{6b}, \psi_{3a}, \psi_{1a}, \psi_{2a}, \psi_{2b}] \\
\psi_{0a} &= \{\textit{bh}, \textit{bhū}, \textit{dhātuaṅga}, \ldots\} \\
\psi_{0b} &= \{\textit{u}, \textit{bhū}, \textit{dhātu}, \textit{aṅga}, \textit{dīrgha}, \textit{udātta}, \delta, \ldots\} \\
\psi_{5a} &= \{\textit{o}, \textit{bhū}, \textit{dhātu}, \textit{aṅga}, \delta, \ldots\} \\
\psi_{6a} &= \{\textit{a}, \textit{av}, \textit{bhū}, \textit{dhātu}, \textit{aṅga}, \textit{hrasva}, \ldots\} \\
\psi_{6b} &= \{\textit{v}, \textit{av}, \textit{bhū}, \textit{dhātu}, \textit{aṅga}, \ldots\} \\
\psi_{3a} &= \{\textit{a}, \text{Ś}\textit{a}\text{P}, \textit{pratyaya}, \textit{hrasva}, \textit{śit}, \textit{pit}, \ldots\} \\
\psi_{1a} &= \{\textit{l}, \textit{l}\text{AṬ}, \textit{pratyaya}, \textit{ait}, \textit{ṭit}, \delta, \ldots\} \\
\psi_{2a} &= \{\textit{t}, \textit{ti}\text{P}, \textit{pratyaya}, \textit{sārvadhātuka}, \textit{pit}, \ldots\} \\
\psi_{2b} &= \{\textit{i}, \textit{ti}\text{P}, \textit{pratyaya}, \textit{sārvadhātuka}, \textit{pit}, \ldots\}
\end{aligned}
$$

Finally we collect all ψ_is not having a δ, i.e. which are not already replaced. This gives us the desired form *bhavati*.

6 `PaSSim` (Pāṇinian Sanskrit Simulator)

In the following we give a brief description of `PaSSim` (Pāṇinian Sanskrit Simulator) we are developing at the University of Heidelberg.[3]

The program aims towards developing a lexicon on Pāṇinian principles. The user enters an inflected word or *pada* and the system furnishes a detailed, step by step process of its generation. It is written in Python™ and consists of the following modules (see Fig. 1):

6.1 `Database`

This module is for inputting, updating, enhancing and organizing the primary database of fundamental components and attributes. The organization of data serves the purpose of incorporating static information of Pāṇinian formulations. For example, *u*Ṅ is stored with *static* attributes *dhātu*, *bhvādi*, *aniṭ* and that its second phoneme is *it* - marker etc. Thus, the effect of many definition rules of Aṣṭādhyāyī are stored in the database. The database is in ASCII and each fundamental component or attribute has a unique key corresponding to which is a (Python-) dictionary.

6.2 `Grammar`

This is the main module. It contains abstract classes corresponding to `SoundSets`, `LanguageComponents` and `ProcessStrips`. Further it has a number of functions like `a61065()`, which simulate the individual rules of Aṣṭādhyāyī.

[3] `http://sanskrit.sai.uni-heidelberg.de`

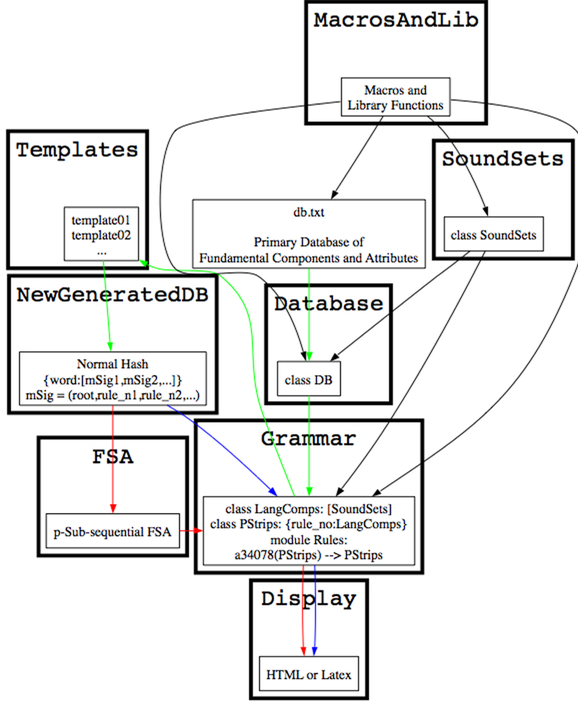


Fig. 1. PaSSim (Pāṇinian Sanskrit Simulator)

6.3 Templates

This module is to organize the process or *prakriyā*. A template prescribes the rules in order of applicability for a group of primary verbs or nominal stems. Templates are specified manually, taking into account the *prakriyā* texts e.g. Siddhānta-Kaumudī [6].[4] It uses Grammar to generate the morpho-syntactic word forms or *pada*s.

6.4 FSA

This module is for the sake of effecient representation of generated words together with the initializing fundamental component(s) and list of rule numbers. These are stored as a p-subsequential transducer [4].[5] The output string associated with a word, thus provides the initializing fundamental components and a list

[4] We would like to acknowledge two texts in Hindi — Vyākaraṇacandrodaya [5] and Aṣṭādhyāyī sahajabodha [2] — which have been very beneficial to us.

[5] Sequential transducers can be extended to allow a single additional output string (subsequential transducers) or a finite number p of output strings (p - subsequential transducers) at final states. These allow one to deal with the ambiguities in natural language processing [4].

of rules. `Grammar` applies these rules one after another and outputs the final as well as intermediate results.

6.5 `Display`

This module provides HTML[6] / LaTeX output. It outputs the content according to the given style sheet for conventions regarding script, color-scheme etc. The phonological, morphological, syntactical and semantical information gathered during the process of generation is rendered in modern terms through a mapping of Pāṇinian attributes corresponding to it.

References

1. von Böhtlingk, O.: Pāṇini's Grammatik. Olms, Hildesheim. Primary source text for our database (1887)
2. Dīkṣita, P.: Aṣṭādhyāyī sahajabodha, vol. 1-4. Pratibha Prakashan, Delhi, India (2006-2007)
3. Katre, S.M.: Aṣṭādhyāyī of Pāṇini. Motilal Banarsidass, Delhi, India (1989)
4. Mohri, M.: On some Applications of Finite-State Automata Theory to Natural Language Processing. Journal of Natural Language Engineering 2, 1–20 (1996)
5. Śāstrī, C.: Vyākaraṇacandrodaya, vols. 1-5. Motilal Banarsidass, Delhi, India (1971)
6. Vasu, S.C., Vasu, V.D.: The Siddhānta-Kaumudī of Bhaṭṭojī Dīkṣita, vols. 1-3. Panini Office, Bhuvanesvara Asrama, Allahabad, India. Primary source text for prakriyā (1905)

[6] See: http://sanskrit.sai.uni-heidelberg.de

Computer Simulation of *Aṣṭādhyāyī*: Some Insights

Pawan Goyal[1], Amba Kulkarni[2], and Laxmidhar Behera[1,3]

[1] School of Computing and Intelligent Systems, University of Ulster, UK
goyal-p@email.ulster.ac.uk,l.behera@ulster.ac.uk
[2] Department of Sanskrit Studies, University of Hyderabad, India
apksh@uohyd.ernet.in
[3] Department of Electrical Engineering, IIT Kanpur, India
lbehera@iitk.ac.in

Abstract. Pāṇini's *Aṣṭādhyāyī* is often compared to a computer program for its rigour and coverage of the then prevalent Sanskrit language. The emergence of computer science has given a new dimension to the Pāṇinian studies as is evident from the recent efforts by Mishra [7], Hyman [5] and Scharf [10]. Ours is an attempt to discover programming concepts, techniques and paradigms employed by Pāṇini. We discuss how the three sūtras: *pūrvatrāsiddham* 8.2.1, *asiddhavad atrābhāt* 6.4.22, and *ṣatvatukor asiddhaḥ* 6.1.86 play a major role in the ordering of the sūtras and provide a model which can be best described with privacy of data spaces. For conflict resolution, we use two criteria: utsarga-apavāda relation between sūtras, and the word integrity principle. However, this needs further revision. The implementation is still in progress. The current implementation of inflectional morphology to derive a speech form is discussed in detail.

Keywords: Pāṇini, *Aṣṭādhyāyī*, Computer Simulation, Conflict Resolution, Event Driven Programming, Task Parallelism.

1 Introduction

Pāṇini's *Aṣṭādhyāyī* is often compared to a computer program for its rigour and coverage of the then prevalent Sanskrit language. It is well known that the *Aṣṭādhyāyī* is not as formal as a computer program. But at the same time, we find many features of modern day computer programming in it. This is a quest to discover programming concepts, techniques and paradigms employed by Pāṇini in writing a grammar for Sanskrit in the form of the *Aṣṭādhyāyī*. In order to do so, we try to 'simulate' the grammar following modern programming practices. We hope, in the process, we may encounter something that may lead to new programming concepts altogether.

As has been rightly pointed out by Scharf[10], it is very much crucial to define what exactly one is going to simulate: is it the *Aṣṭādhyāyī* alone or the *Aṣṭādhyāyī* and *vārtikās* or the grammar as described by Patañjali in his *Mahābhāṣya*?

G. Huet, A. Kulkarni, and P. Scharf (Eds.): Sanskrit CL 2007/2008, LNAI 5402, pp. 139–161, 2009.

At this point in time, we are still open, however our efforts will be to restrict ourselves to the *Aṣṭādhyāyī* as far as possible.

Within the *Aṣṭādhyāyī* itself, sūtras are interpreted in more than one way, there are controversies over the domain of adhikāra sūtras, etc. The sūtras being too many, it is not within the reach of a human being to ensure the consistency with different interpretations. It is therefore a challenge for computer scientists to design a system that simulates the *Aṣṭādhyāyī* and at the same time provide a facility to test the consistency of the whole system for the chosen hypothesis/interpretation.

The paper has been organised as follows: In section 2, we describe the earlier efforts that deal with the implementation and principles of the *Aṣṭādhyāyī*. In section 3, we describe Pāṇini's process from a programming perspective. The role of three sūtras: *pūrvatrāsiddham* 8.2.1, *asiddhavad atrābhāt* 6.4.22, and *ṣatvatukor asiddhaḥ* 6.1.86 in the ordering of the sūtras is discussed. The discussion of these sūtras lead to a model that can be best described with privacy of data spaces. Section 4 deals with the actual implementation, in particular modules for automatic rule triggering and conflict resolution. Challenges, exceptions and problems are described in section 5,6 and 7 respectively. Finally, we discuss future work and give an example of derivation of *rāmāṇām*, genitive plural of the nominal stem *rāma*.

2 Earlier Efforts

There have been attempts to model Pāṇini's *Aṣṭādhyāyī* by Mishra[7] and Scharf[10]. Mishra has proposed a structure for developing a lexicon on Pāṇinian principles. His system shows the rules involved in the morpho-phonemic derivation using manually developed lexicon rich with feature structure. This may serve as a good model to discover different attributes of the lexicon used by Pāṇini, especially the non-formal or extra linguistic features which Pāṇini has used. But this system in its true sense is not the simulation of the *Aṣṭādhyāyī* as it does not give any insight about choice and ordering of the rules in the derivation process.

Scharf has implemented sandhi, nominal declension and verbal conjugation, closely following Pāṇinian rules. However, in his implementation of nominal declension also the rules for different stems are selected and ordered manually. He himself admits that "... it is not a close model of Paninian procedure"[10]. In a sense, his implementation of noun declensions is closer to the arrangement of sūtras in the *Siddhāntakaumudī*[12]. It does not give us a flavour of the Pāṇinian arrangement of rules.

If one decides to simulate the *Aṣṭādhyāyī* on computer, two important questions one needs to answer are:

- How are the rules triggered?
- If more than one rule is triggered then how is the conflict resolved?

Tradition has extensive literature on conflict resolution. The *Paribhāṣenduśekhara* of Nāgeśa Bhaṭṭa discusses many *paribhāṣā*s ('metarules') that are necessary to

understand the interaction of rules, and conflict resolution. The major paribhāṣā dealing with conflict resolution is

paranityāntaraṅgāpavādānām uttarottaraṃ balīyaḥ

Kiparsky[6] explains how the principle of economy (*lāghava*) leads to the concepts of normal blocking and collective blocking which govern conflict resolution.

3 *Aṣṭādhyāyī*: A Programming Perspective

The whole purpose of the Pāṇinian enterprise is to model the communication process. The communication process has two parts – the speaker expressing his thoughts through a language string (generation) and the listener understanding the language string (analysis). Pāṇini's *Aṣṭādhyāyī* models the generation. The process of generation is further assumed to have intermediate levels as in the following figure (Bharati et al.[1]: 63).

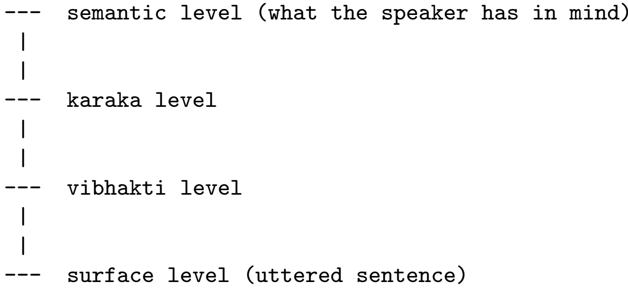


Fig. 1. Levels in the Paninian model

Thus the input for the *Aṣṭādhyāyī* is the semantic level description of thoughts in the speaker's mind in terms of the Sanskrit lexicon, and the output is the sentence in Sanskrit which conveys the same thoughts within the constraints of the language.

The sūtras in the *Aṣṭādhyāyī* are broadly classified into the following six types:[1]

1. *saṃjñā*
2. *paribhāṣā*
3. *vidhi*
4. *niyama*
5. *atideśa*
6. *adhikāra*

[1] *saṃjñā ca paribhāṣā ca vidhir niyama eva ca*
atideśo'dhikāraś ca ṣaḍvidham sūtralakṣaṇam

Most of the rules in the *Aṣṭādhyāyī* are of the type *vidhi*.

Traditionally recognized divisions of the *Aṣṭādhyāyī*, that is, eight *adhyāyas* each divided into four *pādas*, do not, in general, mark off the topics found therein. The topics dealt with in each of the eight adhyāyas are as follows:

- The first adhyāya serves as an introduction to the work in that it contains
  1. The definitions of the majority of the technical terms used in the work,
  2. Most of the metarules, and
  3. Some operational rules.

  The chapter contains rules mainly related to the indicators (*it*) (1.3.2-1.3.9), assignment of *ātmanepada* and *parasmaipada* suffixes (1.3.12-1.3.93), *kāraka* definitions (1.4.23-1.4.55) and *nipātas* (1.4.56-1.4.97).
- The second adhyāya contains four major divisions:
  1. Rules of compounding (2.1.3-2.2.38)
  2. Assignment of cases (2.3.1-2.1.73)
  3. Number and gender of compounds (2.4.1-2.4.31)
  4. *luk* elision (2.4.58-2.4.84)
- The third, fourth and fifth adhyāyas deal with suffixes. The third adhyāya contains the suffixes that are added to a verbal root, while the other two contain the suffixes that are added to a nominal stem.

  In short, the first five adhyāyas build the basic infrastructure that is necessary to set up the proper environment to carry out the derivations smoothly.
- The sixth, seventh and eighth adhyāyas deal with the sūtras that bring about a series of transformations related to continuous text (*samhitā*), word accent, and stem shape.

3.1 Data + Algorithm = Program

The main body of the *Aṣṭādhyāyī* is called the *sūtrapāṭha* (set of rules) and consists of around 4,000 rules. It is accompanied by five ancillary texts: the 14 *pratyāhāra sūtra*s, the *dhātupāṭha* (list of verbal roots), the *gaṇapāṭha* (several collections of particular nominal stems), *uṇādi sūtra*s, and *liṅgānuśāsana* (sūtras describing the gender of different words). Thus we see a clear separation of data (the ancillary texts) from the algorithms (sūtras).

3.2 Data Encapsulation

The third generation languages were based on the fundamental concept of Data + Algorithm = Program. On the other hand, in Object Oriented Programming there is an encapsulation of data and algorithms in the form of objects. Both these aspects are not contradictory to each other. We require the separation of extensive data from algorithms. At the same time in certain cases, we also require the active binding of data with procedures. In Pāṇini's system we notice an intelligent use of both these features. As reported earlier, there is a clear separation of data from algorithms. At the same time, the data in the database consisting of the ancillary texts is encapsulated. All the indicators used by Pāṇini trigger

some functions: For example, the indicators that accompany each root (*dhātu*) in the *dhātupāṭha* mark the dhātu as either ātmanepada or parasmaipada, the *ñi* indicator in verbal roots indicates that such a root takes the suffix *kta* (which is otherwise a past passive participle) in the sense of present tense, as in

ñidhṛṣā + kta - > dhṛṣṭa

3.3 Subroutines

The rules related to a particular task are grouped together. For example, consider the following sūtras which identify sounds used as markers (*anubandha*) in the texts that comprise the grammar.

- *upadeśe aj anunāsika it* 1.3.2
- *hal antyam* 1.3.3
- *na vibhaktau tusmāḥ* 1.3.4
- *ādir ñiṭuḍavaḥ* 1.3.5
- *ṣaḥ pratyayasya* 1.3.6
- *cuṭū* 1.3.7
- *laśakv ataddhite* 1.3.8

If we take into account the recurrence (*anuvṛtti*) of terms from preceding sūtras, the rules may be rewritten (indicating the anuvṛtti by indentation) as

- *upadeśe*
  - *ac anunāsika (=) it* 1.3.2
  - *hal antyam 1.3.3*
    - *na vibhaktau tusmāḥ (=it)* 1.3.4
  - *ādiḥ*
    - *ñiṭuḍavaḥ (=it)* 1.3.5
    - *pratyayasya*
      - *ṣaḥ (=it)* 1.3.6
      - *cutū (=it)* 1.3.7
      - *laśaku (=it) ataddhite* 1.3.8

Translation of this set of rules into a simple algorithm will show the parallel between Pāṇini's sūtras and a computer algorithm.

```
if(input is from UPADE‘SA)
     Mark the ANUNASIKA AC as INDICATOR
  if(last var.na(X) is HAL)
     if(the input is neither VIBAKTI nor TUSMA)
        Mark X as INDICATOR
     endif
  endif
  if(the beginning syllable(Y) is ~ni or .tu or .du)
     Mark Y as INDICATOR
  endif
  if(the input is a PRATYAYA)
```

```
    if(Y is .sa)
       MARK Y as INDICATOR
    endif
    if(Y is from ca_varga or .ta_varga)
       Mark Y as INDICATOR
    endif
    if(the input is NOT TADDHITA)
       if(Y is either la or ‘sa or ka_varga)
          Mark Y as INDICATOR
       endif
    endif
endif
```

There are many such instances of well-defined subroutines spread all over the *Aṣṭādhyāyī*.

3.4 Operations

The nature of the problem the *Aṣṭādhyāyī* deals with indicates that the typical operations involved are various kinds of string operations. They are of four types.

- assigning a name
- substitution
- insertion
- deletion

It has been already recognized ([4], [2]) that Pāṇini expresses all such rules as context sensitive rules. He ingeniously uses cases (*vibhakti*s) to specify the context. A typical context sensitive rule is of the form

$$\alpha\beta\gamma \Rightarrow \alpha\delta\gamma$$

Pāṇini uses 5^{th} and 7^{th} case to indicate the left and right context respectively, 6^{th} case to indicate the element that will undergo a change and 1^{st} to indicate what it will change to. Here is an example from the *Aṣṭādhyāyī*.

ato ror aplutād aplute (ut ati saṃhitāyām) 6.1.113
at{5} ru{6} apluta{5} apulta{7} (ut{1} at{7})
non-pluta_at **ru** *non-pluta_at* $\Longrightarrow$ *non-pluta_at* **ut** *non-pluta_at*
gloss: change the ‘ru’ preceeded and followed by an ‘a-pluta a’ to ‘u’.

3.5 Ordering of the Rules

Three sūtras in the *Aṣṭādhyāyī* play an important role in deciding the order of the rules. These sūtras are

- *pūrvatrāsiddham* 8.2.1
- *asiddhavad atrābhāt* 6.4.22
- *ṣatvatukor asiddhaḥ* 6.1.86

We discuss each of these sūtras and show how they govern the ordering.

Asiddham. Traditionally the *Aṣṭādhyāyī* is divided into 2 parts – *sapāda-saptādhyāyī* and *tripādī*. The rule *pūrvatrāsiddham* (8.2.1) makes the output of the rules in the latter part unavailable to the earlier rules. Further this rule being the adhikāra sūtra makes the output of each of the following sūtras unavailable to the previous rules within the *tripādī*. This necessarily implies that the *tripādī* should follow the *sapāda saptādhyāyī* and that the rules within the *tripādī* should also be followed linearly or sequentially. Based on this, it is very likely that one would be tempted to model *tripādī* as a single subroutine, where all the rules are applied sequentially and the intermediate output is stored in local variables.

But then it would not be a faithful representation. Pāṇini did not state it this way. The sequential ordering of sūtras in the *tripādī* and application of *tripādī sūtra*s after the application of rules from the *sapāda saptādhyāyī* is an inference we draw from the adhikāra sūtra. Instead of inferring, let us try to understand precisely what Pāṇini has said. The word *asiddham* means – (regarded as) not existing ([8], p. 120, col. 3). To give an analogy, in programming paradigm, the variables local to a subroutine are regarded as not existing(or not visible) with respect to the calling function. We model the sūtra *pūrvatrāsiddham* as follows: The result of applying the sūtras in this section should not be available to the earlier sūtras in the *sapāda saptādhyāyī*. Similarly, the sūtra being the adhikāra sūtra, the result of the later sūtra in the *tripādī* should also be not available to the earlier sūtras in the *tripādī*. It essentially means that each of the sūtras in the *tripādī* section should have its own data space and the data space of the later sūtras be invisible to the earlier sūtras. Thus this model does not implement *tripādī* as a single subroutine, but keeps each of the rules (or a group of rules forming a subroutine) as a single separate unit. The invisibility of the data spaces of later rules to the earlier rules ensures that the rules are applied only sequentially.

Asiddhavat. Within the *sapāda saptādhyāyī* there is a section known as the *asiddhavat* section. The sūtra

asiddhavad atrābhāt 6.4.22

is translated by Vasu([13]) as

"The change, which a stem will undergo by the application of any of the rules from this sūtra up to 6.4.129 is to be considered as not to have taken effect, when we have to apply any other rule of this very section 6.4.23-6.4.129".

As an example, let us consider the derivation of *śādhi* from *śās + hi*.
Two sūtras

hujhalbhyo her dhiḥ 6.4.101

and

śā hau 6.4.35

are applicable.
6.4.101: *śās* + **hi** ⇒ *śās* + **dhi**
6.4.35: **śās** + *hi* ⇒ **śā** + *hi*

As is evident from this, if 6.4.101 is applied, then the conditions for applying 6.4.35 are not met and hence it would not be applicable. Similarly, if 6.4.35 is applied first, then the conditions for 6.4.101 would not be met and it would not be applicable. The word *asiddhavat* means 'as if it is not applied'. So after applying 6.4.35, though *śās* changes to *śā*, still the result is not visible to 6.4.101 and hence 6.4.101 changes *hi* to *dhi*. As a result of both these rules, *śās* + *hi* changes to *śādhi*. Thus instead of stating the rule as

$$\text{R: a b} \Rightarrow \text{c d}$$

Pāṇini states it as a combination of two rules:

$$R_1 : ab \Rightarrow cb$$

$$R_2 : ab \Rightarrow ad$$

and thereby one may conclude that Pāṇini achieves economy.[2] However, if one looks at the complete asiddhavat section, one finds only handful of examples that require parallel application of rules, and hence it is not worth stating that economy is achieved. Nevertheless, it provides an example of task parallelism. Further this also has an impact on parameter passing. Since the same input should be available to all the rules in this section, the input should be passed as a value. But at the same time, a local copy of it will undergo necessary changes. One or more processes run in parallel, and the consolidated result of all these processes is then passed back to the calling function.

Though this interpretation also seems to be consistent with what is said in the sūtra, still it is also an inference. Pāṇini never mentions that the sūtras are to be applied in parallel (Bronkhorst[3]). He uses the term *asiddhavat*. So to be faithful to the Pāṇini's system then, the results of the application of sūtras in this section should not be visible to other sūtras of the same section. This is possible, if we assign a separate data space to the rules in this section, which is not visible to the rules within this section.

Asiddhaḥ. The third type of *asiddha* is provided by the sūtra

ṣatvatukor asiddhaḥ 6.1.86

This rule, which occurs under the adhikāra *ekaḥ pūrvaparayoḥ* 6.1.84 in effect up to 6.1.110, says that the single replacement (*ekādeśa*) that will result through the application of rules under this adhikāra, is *asiddha* with respect to the two processes, viz. *ṣatva* and *tuk*. That is, the result of application of rules in the *ekādeśa* section is invisible to the rules which correspond to the *ṣatva* or *tuk* processes. Thus here again, there is a concept of data space, the result of the operations in the *ekādeśa* section are written to this data space, which are unavailable to the rules performing *ṣatva* and *tuk* operations.

[2] If there were n_1 rules of type R_1 and n_2 rules of type R_2, then there would have been $n_1 * n_2$ (possible combinations) rules of type R. However, by making them applicable in parallel there are only $n_1 + n_2$ rules, and thus economy is achieved.

3.6 Programming Model

The typical grammarians view of the *Aṣṭādhyāyī* may be stated as follows:

'The rules in the *sapāda saptādhyāyī* seek for an opportunity to act on an input by finding conditions (*nimitta*) in which they are applicable. In case there is a conflict, there are certain conflict resolution techniques (described as *paribhāṣā*), which come into play. The conflict resolver selects one rule and effects changes in the data space.'

This model is described in the Figure 2.

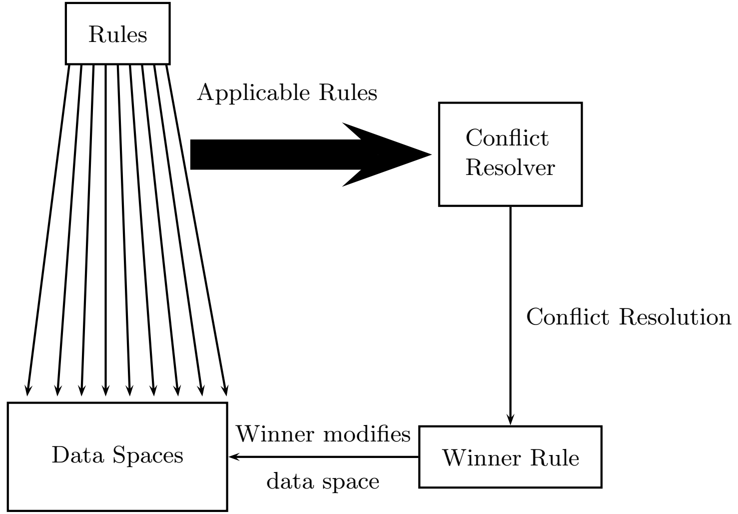


Fig. 2. Application of rules

Thus we notice a striking similarity between the event driven programming and triggering of *Aṣṭādhyāyī* rules. The *nimitta* or the context triggers an appropriate rule. The saṃjñā sūtras for example, assign different 'attributes' to the input string, thereby creating an environment for certain sūtras to get triggered. The adhikāra sūtras assign necessary conditions to the sūtra for getting triggered. Paribhāṣā sūtras provide a meta language for interpreting other sūtras. The niyama and atideṣa sūtras put restrictions on or extend the domains in which the sūtras are to be applied. Finally it is the vidhisūtras which effect the transformations.

The rules fall under four categories: *asiddham*, *asiddhavat*, *asiddha* and the rest[3]. Each of these rules has its own data space where it writes its own output

[3] This model is an improvement over the one emerged while teaching a course on 'Structure of Aṣṭādhyāyī at Tirupati, in 2006-07 (described in 'Computational Structure of Aṣṭādhyāyī and Conflict Resolution Techniques'(Subbanna S. and Varakhedi S., 2009)): the major shift is in modelling with private data spaces. The interactions of the rules belonging to *ṣatva* and *tuk* sections with the other rules is also made explicit. The current model based on privacy of data spaces has its seed in the discussions Amba Kulkarni had with Vineet Chaitanya.

Table 1. Affected Data Spaces

Rule	Affected Data Space
siddha	D_0
asiddhavat	D_1
asiddhaḥ (rules in ekādeśa section)	D_2
asiddham (*Tripādī - ṣatva*)	$D_{8.2.1}$ $D_{8.2.2}$ $D_{8.2.3}$ . . $D_{8.4.68}$

Table 2. Visible Data Spaces

Rule	Visible Data Spaces
Rest	D_0, D_1, D_2
asiddhavat	D_0, D_2
ṣatva and tuk	D_0, D_1
asiddham(*Tripādī - ṣatva*)	D_0, D_1, D_2, $D_{8.2.1}$
8.2.2	D_0, D_1, D_2, $D_{8.2.1}$, $D_{8.2.2}$
8.2.3	D_0, D_1, D_2, $D_{8.2.1}$, $D_{8.2.2}$, $D_{8.2.3}$
.	.
.	.
8.4.68	D_0, D_1, D_2, $D_{8.2.1}$,..., $D_{8.4.68}$

(see Table 1). The visibility of these data spaces to different categories of rules is described in Table 2.

Now we illustrate, with the help of examples, how the privacy of data spaces lead to the correct generation in case of *asiddhavat*, *asiddham* and *asiddhaḥ* sections.

– *asiddhavat* Consider the derivation of *śādhi*: imperative second person singular form of the root *śās*. The derivation of our interest starts with the data space D_0 having *śās* + *hi* in it. Now two sūtras, viz. *hujalbhyo her dhiḥ*(6.4.101) and *śā hau*(6.4.35) are applicable. The constraint resolver returns both the rules. Thus there are two different orders of applying these rules. Let us assume that 6.4.101 is applied first. Then *hi* changes to *dhi*. This result will be available in the data space D_1 which is not visible to sūtra 6.4.35. As such, 6.4.35 applies on the contents of the data space D_0 and changes *śās* to *śā*. This change also gets stored in D_1. Thus now D_1 contains *śā* + *dhi*. As is clear, even if the order of application of sūtras is changed, we get the same result as in D_1.

  Consider another derivation: *jahi*, from the root *han*. The step in the derivation in which we are interested is when the data space D_0 has *han* + *hi* in it. *hanter jaḥ* (6.4.36) changes the input to *ja* + *hi*. Since this sūtra

is from the *asiddhavat* section, the output is written to data space D_1, and hence is not visible to any other rule in the *asiddhavat* section; in particular to *ato heḥ* (6.4.105). Therefore there is no question of *ato heḥ* getting triggered.

– *asiddhaḥ* Let the data space D_0 has *adhi* + *i* + *lyap*. Now two sūtras, viz. *akaḥ savarṇe dīrghaḥ* (6.1.101) and *hrasvasya piti kṛti tuk* (6.1.71) are applicable. With the data space model, we explain how the order of the rules gets fixed automatically. Let us assume that rule 6.1.101 is applied first. Since this rule belongs to the *ekādeśa* section, the result, viz. *adhī* + *lyap* will be stored in the data space D_2, and hence will not be visible to 6.1.71, as such the latter rule operates on the data in D_0, and changes it to *adhi* + *i* + *tuk* + *lyap*. Again at this stage, 6.1.101 can see the contents of D_0, and hence changes it to *adhī* + *tuk* + *lyap*. Had we applied 6.1.71 first and then 6.1.101, it would have resulted in the same string. Or in other words, the order 6.1.71 and then 6.1.101 is optimal, than 6.1.101, followed by 6.1.71 followed by 6.1.101 again. Application of 6.1.101 before 6.1.71 being redundant, implementation of preferred order is very straight forward.

  Consider another example. The input of our interest is *ko* + *asicat* stored in the data space D_0. *eṅaḥ padāntād ati* (6.1.109) changes it to *kosicat*. This result is stored in the data space D_2. As such this result is not visible to the *ādeśapratyayoḥ* (8.3.59) from the *ṣatva* section and thus 8.3.59 is not applicable. Thus the undesirable result *koṣicat* is not generated.

– *asiddham* Finally take the example of *vac* + *ti*. Two sūtras, viz. *coḥ kuḥ* (8.2.30) and *stoḥ ścunā ścuḥ* (8.4.40) are applicable. Since the result of 8.4.40 (stored in $D_{8.4.40}$) is not visible to 8.2.30, application of 8.4.40 before 8.2.30 will be redundant. After application of 8.2.30, there will not be any scope for 8.4.40, giving the desired result *vakti*.

Thus, these examples illustrate how the concept of data spaces represent the simulation of *Aṣṭādhyāyī* faithfully.

4 Implementation

Simulation of the *Aṣṭādhyāyī* involves the following factors:

1. Interpretation of sūtras using the metalanguage described by Pāṇini in the *Aṣṭādhyāyī*,
2. Faithful representation of sūtras,
3. Automatic triggering of rules, and
4. Automatic conflict resolution.

In this paper we have concentrated only on 3. For conflict resolution we have used two main principles: An *apavāda* rule and a rule belonging to the *aṅga* section are given priority over the others. Regarding the representation of sūtras, we use regular expressions to represent the patterns, and positions to represent the left and right context, the string that undergoes a change and the resultant string.

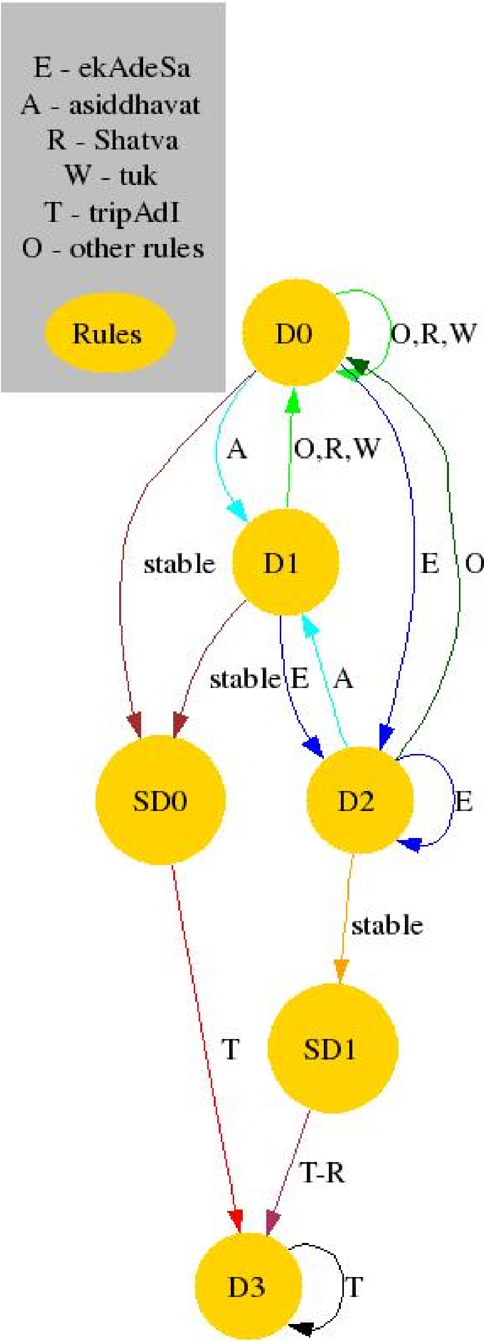


Fig. 3. Modelling Aṣṭādhyāyī

The model of the *Aṣṭādhyāyī* that we are implementing is presented in the Figure 3.

As the diagram shows, we have classified the rules in the following way:

1. E - Rules belonging to *ekādeśa*
2. A - Rules belonging to *asiddhavat*
3. R - Rules belonging to *ṣatva vidhi*
4. W - Rules belonging to *tuk vidhi*
5. T - Rules belonging to *tripādī*, excluding *ṣatva vidhi*.
6. O - All other rules

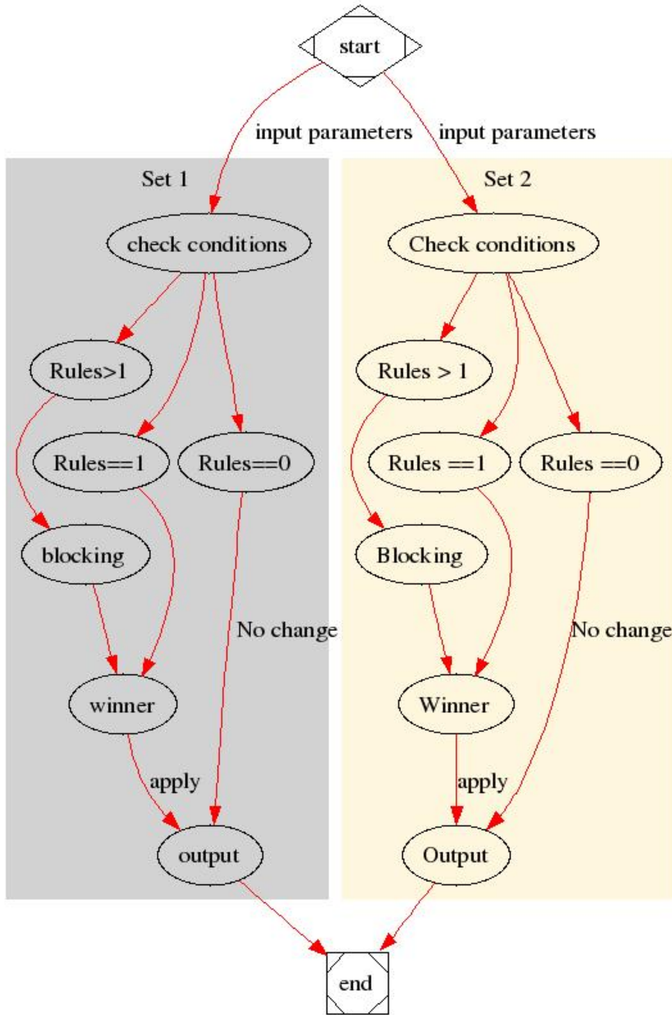


Fig. 4. Parallel execution in asiddhavat

The diagram shows the data flow from one data space to another by the invocation of different types of rules. A rule represented by an arrow takes input from the data space at the tail of the arrow, and writes the output to a data space indicated by the head of the arrow. In the beginning, the input string is stored in the dataspace D_0. A rule from O section can see the contents of only three data spaces viz. D_0, D_1 and D_2. At any stage, when a rule from O section is applicable, among these three data spaces, the data space with latest information is chosen as an input to the rule. If D_0 is the input data space, the output is written to itself. But if D_1 has the latest data, then the output is written to D_0. In case D_2 has the latest data, then the output is written to D_0.

We illustrate this with an example from asiddhavat section.

Let D_0 contain *han* + *hi*, and suppose it has the latest data among all the data spaces. Now the rule *hanter jaḥ* 6.4.36, which is from the asiddhavat section (A) is applicable. This rule then takes the latest data (in this case from D_0), and writes the output to D_1. So D_1 contains *ja* + *hi*, whereas D_0 still continues to have *han* + *hi* in it. The rule *ato heḥ* 6.4.105 can see only D_0 and D_2, and can't see D_1. Hence, 6.4.105 will not be applicable, thereby the wrong generation is stopped. Similarly when the rules from the *ekādeṣa* section are triggered, they write the output to the data space D_2. When no more rules are triggered, then the system enters into a stable state SD_0. This stable state is different (SD_1) if the system is in the state corresponding to *ekādeṣa*. While in one of the stable states, the rules in the *tripādī* section get triggered sequentially. If the system is in the *ekādeṣa* state, and no rule from *tripādī* gets triggered, then the rules from *ṣatva* section also do not get triggered. But if the rules in *tripādī* have been applied, then *ṣatva* rules may get triggered.

If more than one rule from the *asiddhavat* section is triggered, then their behaviour is shown in the Figure 4.

We have started the implementation for getting *śabdarūpa* of nominals given the *prātipadika*, along with the *vacana*, *liṅga*, and *vibhakti* for which we need to decline.

4.1 Hierarchy

We take the input as a sentence. For instance, on input *rāma(puṁliṅgam +ekavacanam+kartṛ) vana(napuṁsakaliṅgam+ekavacanam+karma) gam (laṭ+kartari)*, the machine should produce *rāmaḥ vanam gacchati*, along with the trace of an algorithm showing the exact sequence of the applied rules, and conflict resolution, if any. A sentence has padas as its children. Each pada will have a root word along with the attributes. We will first run the program on the leaf nodes and will keep on merging these together until we reach the sentence. Thus for the example taken, we have the following children in the beginning:

- *rāma(puṅliṅa+ekavacana+kartā)*
- *vana(napuṅsakaliṅa+ekavacana+karma)*
- *gam(laṭ+kartari)*

Each child has a root word along with the attributes. Henceforth we call this structure 'input DS'. We pass the children one by one through the program. Finally, we have different padas, and that will be the output (*rāmaḥ vanam gacchati*).

4.2 Structure of Input DS

An input DS contains an array of words along with their attributes. It allows the addition of a particular attribute as well as removal if necessary. We have allowed substitution as well as augmentation in the structure. Let us briefly discuss these two operations:

Augmentation. The augmentation process needs two parameters: a string to be augmented and the position where it is to be augmented. The function will look at the rules applicable (such as *ādyantau ṭakitau* 1.1.46 and *midaco'ntyāt paraḥ* 1.1.47), and the change is effected at appropriate position in the input DS.

Substitution. Substitution is also permitted within the input DS. The parameters specified are the same as in augmentation. The rules such as *ṅic ca* 1.1.53, *anekālśit sarvasya* 1.1.55, *ādeḥ parasya* 1.1.54 and *alo'ntyasya* 1.1.52 are the governing rules out of which one is applied depending upon the input DS and after the substitution the DS is returned.

4.3 Structure of a Rule

A typical vidhi rule of the *Aṣṭādhyāyī* has a context sensitive form:

$$\alpha\beta\gamma \Rightarrow \alpha\delta\gamma$$

α and γ specify the context, β is the domain which undergoes change and δ is the resultant/changed part of the string.
Corresponding to each rule now we require three functions:

1. a function that checks whether the rule is applicable or not,
2. a function to compute the result if the function is applicable, and finally
3. a function that returns the conditions under which the rule is applicable.

Thus:

1. Each rule has been stored as a structure involving patterns that stores the conditions under which the rule is applicable.
2. Then we apply pattern matching to come up with the list of rules triggered for a given input DS. The application part has been stored separately, which takes input as the rule number of the winner rule and changes the input DS, accordingly.
3. In case there is more than one rule applicable, then the conflict resolution module looks at the conditions for each of the applicable rules and chooses the "winner rule", after resolving the conflict if any. Conflict resolution is discussed in the next section.

4.4 Modules

Each *prakṛti* as well as the *pratyaya* part of the *pada* passes through a series of modules. The modules are of 2 types. Some modules just look at the input and assign different names (*saṃjñās*) to the different parts of the input string; others transform the input string.

Module for assigning saṃjñās to the prakṛti. A prakṛti is assigned an appropriate saṃjñā using the

– databases, or
– current state of the input string.

For example, a prakṛti gets the saṃjñā *dhātu* if it is found in the *dhātupāṭha*, the saṃjñā *sarvanāma* if it is found in the gaṇa *sarvādi*, etc. Sometimes some pratyaya creates a context for certain saṃjñās such as *bha, ghi, nadī*, etc.

it **Module:** The dhātus in the *dhātupātha*, pratyayas, etc. are given with some markers commonly called *anubandha*s and termed *it* by Pāṅini. These markers need to be identified and marked, before the processing starts. The module takes as input a particular *upadeśa* and marks the varṇa as *it*. The rules 1.3.2 to 1.3.8 described in section 3.3 mark the *it*s. Finally, the phonemes marked *it* are deleted from the input string by the rule *tasya lopaḥ* 1.3.9. However the markers are stored in the DS, as they actively bind the procedures.

Module for *pratyaya vidhi*. In this module, a pratyaya undergoes some changes based on the characteristics of the *aṅga*[4] and pratyaya. All the rules of this particular vidhi are listed as an array 'pratyaya vidhi rule' in a particular data structure as follows:

1. *aṅga ending*: Denotes the context of the endings of an *aṅga*.
2. *aṅga attrib*: Represents the attributes (characteristics) of the *aṅga*.
3. *pratyaya*: If there are some special contexts for the pratyaya.
4. *pratyaya attrib*: Represents the attributes of the pratyaya.
5. *rule number*: Keeps the information of the rule number.

The data structure has been made in Java, and constructors are made to directly encode the conditions of a particular *pratyaya vidhi* rule. For example, the rule *ato bhisa ais* has been encoded as:

```
list_pratyaya_vidhi_rule[0]=new pratyaya_vidhi_rule(''7-1-9'',
''ataH bhisaH ais'', ''a'', '''', '''', ''root(bhis)'');
```

[4] *In accordance with yasmāt pratyayavidhis tadādi pratyaye 'ṅgam* 1.4.13, the *prakṛti* to which an affix has been provided, is termed *aṅga*.

The '[0]' indicates this is the first rule of the array. The rule constructs a 'pratyaya vidhi rule' which needs the ending of the aṅga to be *a*, and the pratyaya to be the one with its original form as *bhis*. The rule number too has been stored. All the rules of *pratyaya vidhi* are stored in a similar fashion. When a particular 'input DS' is passed through this array, another array 'triggered pratyaya vidhi rule' enlisting all the rules which are applicable is produced. Finally, the conflict resolver decides the 'winner rule".

After the winner rule has been selected, the subroutine 'apply' is called. For example, the above rule substitutes *ais* for the case ending *bhis* after a prātipadika ending in *a*. The input DS is passed through the substitution subroutine with the information that *ais* is to be substituted. Thus, for the input DS with (*rāma*(attributes)+*bhis*(attributes)), the rule 7.1.9 will apply as 'substitute(input DS, *ais*, 1)', where we pass the information that substitution has to be done at index 1. The result of this substitution will be: (*rāma*(attributes) +*ais*(attributes)).

Module for *aṅga vidhi*. In this module, an aṅga undergoes some changes based on the characteristics of the *aṅga* and pratyaya. The rules are listed in the same data structure as used for rules belonging to *pratyaya vidhi*. The rules are stored in an array 'list anga vidhi rule'. Consider the encoding of the rule *aco ñṇiti* which states, "Before the affixes having an indicatory *ñ* or *ṇ*, *vṛddhi* is substituted for the end vowel of a stem".

```
list_anga_vidhi_rule[9]=new anga_vidhi_rule(''7-2-115'',
"acaḥ ñṇiti", ac,"","",''N-it|~N-it'');
```

This is the tenth rule of the array corresponding to *aṅga vidhi* rules. From the encoding, we can infer the conditions that the aṅga should be ending in a vowel (*ac*) and the pratyaya should have either *ṇ* or *ñ* as a marker.

After passing the input DS through the array, we will get the array 'triggered anga vidhi rule' upon which the conflict resolution module will be called giving the winner rule. Finally the subroutine 'apply' will act. Consider the formation of the nominative singular (*prathamā ekavacana*) of the root *go*. Since the pratyaya is termed *sarvanāmasthāna*, it gets the characteristics of indicatory *ṇ* by the rule *goto ṇit* 7.1.90, a rule in *pratyaya vidhi*. In *aṅga vidhi*, the winner rule will be 7.2.115 and it will do *vṛddhi* on the end vowel of *go* giving the structure as (*gau*(attributes)+*as*(attributes)).

Module for *asiddhavat*. In the *asiddhavat* section, we have certain rules belonging to both *pratyaya vidhi* and *aṅga vidhi*. The speciality of this section is that a rule in this section does not see the changes made by other rules in the same section. To implement this, the rules have been grouped in two separate arrays, belonging to *pratyaya vidhi* and *aṅga vidhi*. The same input string is passed to these arrays. The *pratyaya vidhi* rules may change the pratyaya, and the *aṅga vidhi* rules may change the aṅga. The aṅga from the *aṅga vidhi* module and the pratyaya from the *pratyaya vidhi* module are taken together to the input

DS which is available to other rules. Let us consider the formation of *śādhi*. The input DS that is passed to this section is (*śās*(attributes) + *dhi*(attributes)). We are passing the same copy to both the *aṅga vidhi* and the *pratyaya vidhi* modules of the *asiddhavat* section. In each of the two *vidhi* modules, if more than one rule is triggered, the conflict resolution module selects one winner for each type of the rules separately. The module is described in the Figure 4.

Module for Sandhi. We have enlisted all the rules belonging to sandhi in an array 'list Sandhi rule'. The encoding format is the same as the one used for the *pratyaya vidhi* and *aṅga vidhi* modules. For example, the conditions for the rule *eco'yavāyāvaḥ* 6.1.83 have been stored as:

```
list_Sandhi_rule[1]=new Sandhi_rule
(``6-1-78'',``(eco)83 ayavAyAvaH'',ec,ac);
```

In this case, we have constructors which build the object 'Sandhi rule' with the information of the last letter of the first word and initial letter of the second word. The rule states, "The vowels belonging to the pratyāhāra *ec*, are replaced by *ay, āy, av, āv* respectively provided the second word starts with a vowel." The processing is the same as discussed in the *pratyaya vidhi* section.

Module for *tripādī*. In the rules belonging to this section, we need not have a conflict resolution module since rules are applied in linear order. So, we need only to check the conditions and apply the rule if the condition is satisfied. This module is visited after we are sure that the output has become stable after going through the other modules.

Module for Conflict Resolution: The module is made independent of the category of the rules. It takes as input two rules at a time and returns the rule that blocks the other (it returns the superior rule). Let the two rules be Rule A and B. If the domain of rule A is properly included in the domain of rule B, then A blocks rule B. While applying blocking, we look at the following properties in the decreasing order of their priority:

1. Whether there is conflict (*pratiṣedha*) in the rules: If the two rules present no pratiṣedha, i.e. application of rule A doesn't bleed (depriving of conditions) rule B, and A and B are in the order of the *Aṣṭādhyāyī*, we apply the rules in order.
2. Word integrity principle: If a rule from *ekādeśa* interacts with a rule from the *aṅga vidhi* module, the rule from the *aṅga vidhi* module is the winner using this principle.
3. The environment-changing rule is given precedence over the rule that is not environment-changing. Thus if rule A bleeds rule B, but rule B doesn't bleed rule A, rule A is the winner due to changing the environment.
4. Whether a rule is specific to a particular initially taught item: If a rule specifies a particular initially taught item, it is preferred over other rules. For example:

We consider the formation of the accusative singular (*dvitīyā ekavacana*) of the base *rāma*. When it passes through the sandhi module, it has the structure (*rāma*(attributes) + *am*(attributes)). All the rules in the sandhi module check for the applicability conditions and the 'triggered Sandhi rule' array contains the following rules:
(a) *ād guṇaḥ* 6.1.87
(b) *akaḥ savarṇe dīrghaḥ* 6.1.101
(c) *prathamayoḥ pūrvasavarṇaḥ* 6.1.102
(d) *ato guṇe* 6.1.97
(e) *ami pūrvaḥ* 6.1.107
From the above stated rules, the rule *ami pūrvaḥ* is applicable only when the second item has *a* belonging to the initially introduced item *am*. This initially introduced item restricts the domain for this rule and it is preferred over other rules.
5. Whether a rule is specific to a particular feature of a domain: Consider the formation of the dative plural (*caturthī bahuvacana*) of the base *rāma*. When it passes through the *aṅga vidhi* module, it has the structure (*rāma*(attributes) + *bhyas*(attributes 'bahuvacana')). After the condition checking the 'triggered anga vidhi rule' array contains the following rules:
(a) *supi ca* 7.3.102
(b) *bahuvacane jhaly et* 7.3.103
Out of the above two rules, rule 7.3.103 requires a special property of *bahuvacana*, which makes it preferable over the rule 7.3.102.
6. Whether a rule is applicable for fewer sounds (*al*): A rule involving fewer phonemes in the context is given more priority over the one involving more.

5 Challenges

Whether to include vārtikās?

In the traditional view, there are vārtikās that handle those cases where the rule *vipratiṣedhe param kāryam* or the paribhāṣā *paranityāntaraṅgāpavādānām uttarottaram balīyaḥ* doesn't give the right result. The question arises whether it is necessary to go for these vārtikās or we can resolve the conflicts without resorting to the vārtikas. We enlist one of the cases below:

Formation of *vāri* + *ṅe*. We are at the structure *vāri(napuṃsaka)+e(ṅ-it, sup)*. The rules that are triggered are:

– *iko'ci vibhaktau* 7.1.73
– *gher ṅiti* 7.3.111

Rule 7.3.111 is the later rule and should be applied according to the principle that a later rule takes precedence over an earlier one. However, it does not apply; 7.1.73 applies instead. For this there is a vārtikā *vṛddhyautvatṛjvadbhāvaguṇebhyo num pūrvapratiṣedhena* which says that the later rule is not applicable when *num* is ordained by a previous rule and one of *vṛddhi, autva, tṛjvadbhāva* or *guṇa* is ordained by a later rule. Till the implementation of the algorithm presented here, we do not have a satisfactory answer for this.

6 Exceptions

There are certain exceptions in the *Aṣṭādhyāyī* which need to be handled separately. Consider the formation of *śivorcya.*

- We have the form *śivas* + *arcya.* No rules from the *sapāda saptādhyāyī* are applicable.
- The rule *sasajuṣo ruḥ* 8.2.66 finds scope and is applied. Thus we have the form:

śiva + *ru* + *arcya*

- After the *u* in *ru* will be marked as a marker(*anubandha*), leading to its deletion (*lopa*).

śivar + *arcya*

- The rule *bhobhagoaghoapūrvasya yo'śi* 8.3.17 finds scope and will change the structure to

śivay + *arcya*

which is an *aniṣṭa* form. The *apavāda* of this rule, *ato ror aplutād aplute (ut ati saṃhitāyām)* 6.1.113 gets *ru* in *śivar* by the *sthānivadbhāva* and checks the application of 8.3.17. 6.1.113 changes the *r* to *u* giving the structure:

śiva + *u* + *arcya*

- The rule *ād guṇaḥ* 6.1.87 is applicable giving the form:

śivo + *arcya*

- The rule *eṅaḥ padāntād ati* 6.1.109 is applicable giving the form:

śivorcya

Thus we clearly see that this is an exception for the adhikāra sūtra *pūrvatra asiddham.*

7 Problems

Consider the formation of *ramā* + *ṅe*. The structure is:

ramā + *e*

The rule *yāḍāpaḥ* - 7.3.113 which sees the *ṅit* of the *pratyaya* and does the *āgama* of *yāṭ*, which being *ṭ-it*, sits in the front, and we have

ramā + *yai*

The problem comes that the *ṅit* attribute is still there in *yai* and the rule gets triggered again and again giving the form *ramā+yāyā...yai.* We need to seek solution for this.

8 Future Work

It is necessary to understand how Pāṇini's *Aṣṭādhyāyī* resolves the conflicts. The current implementation is still primitive and not satisfactory. Pāṇini has not mentioned any conflict resolution rules explicitly, but it seems he assumed them implicitly. In the current implementation, the rules are represented using manually coded patterns. It will be interesting to see if the machine can interpret the rules automatically based on the vibhaktis and the meta rules. What difference the *yoga vibhāga* makes in the form of output of conflict resolution will an interesting issue to explore.

References

[1] Akshar, B., Chaitanya, V., Sangal, R.: Natural Language Processing: A Paninian Perspective. Prentice Hall of India, New Delhi (1995)
[2] Bhate, S., Kak, S.: Panini's grammar and Computer Science. Annals of the Bhandarkar Oriental Research Institute 72, 79–94 (1993)
[3] Bronkhorst, J.: Asiddha in the Aṣṭādhyāyī: A misunderstanding among the traditional commentators. Journal of Indian Philosophy 8, 69–85 (1980)
[4] Cardona, G.: On translating and formalizing Paninian rules. Journal of Oriental Institute, Baroda 14, 306–314
[5] Hyman, M.D.: From Paninian Sandhi to Finite State Calculus. In: Huet, G., Kulkarni, A., Scharf, P. (eds.) Sanskrit CL 2007/2008. LNCS (LNAI), vol. 5402, pp. 253–265. Springer, Heidelberg (2009)
[6] Kiparsky, P.: On the architecture of Pāṇini grammar. The lectures delivered at the Hyderabad Conference on the Architecture of Grammar (2002)
[7] Mishra, A.: Simulating the Paninian system of Sanskrit Grammar. In: Huet, G., Kulkarni, A., Scharf, P. (eds.) Sanskrit CL 2007/2008. LNCS (LNAI), vol. 5402, pp. 127–138. Springer, Heidelberg (2009)
[8] Monier Williams, M.: A Sanskrit-English Dictionary. Clarendon, Oxford (1872) (reprint: Motilal Banarasidass, Delhi, 1997)
[9] Roy, P.V., Haridi, S.: Concepts, Techniques and Models of Computer Programming. MIT Press, Cambridge (2004)
[10] Scharf, P.M.: Paninian Grammar. In: Huet, G., Kulkarni, A., Scharf, P. (eds.) Sanskrit CL 2007/2008. LNCS (LNAI), vol. 5402, pp. 95–126. Springer, Heidelberg (2009)
[11] Subrahmanyam, P.S.: Pāṇinian Linguistics, Institute for the Study of Languages and Cultures of Asia and Africa, Tokyo University of Foreign Studies, Japan (1999)
[12] Vasu, S.C.: Siddhānta Kaumudi. Motilal Banarasidas Publishers, New Delhi (2002)
[13] Vasu, S.C.: The Aṣṭādhyāyī of Pāṇini. Motilal Banarasidas Publishers, New Delhi (2003)
[14] Subbanna, S., Varakhedi, S.: Conflict Aṣṭādhyāyī and Conflict Resolution Techniques. In: Kulkarni, A., Huet, G. (eds.) Sanskrit Computational Linguistics. LNCS, vol. 5406, pp. 56–65. Springer, Heidelberg (2009)

Appendix

We illustrate the formation of *rāmāṇām*, the plural, masculine form in genitive case of the root word *rāma*.

1. The input to the program is: Form: *rāma: bahuvacana, ṣaṣṭhī, puṁliṅga.*
2. *arthavad adhātur apratyayaḥ prātipadikam* 1.2.45
   rAma gets the saṃjñā *prātipadika* after being checked in the database, and we have the form: *rāma (**prātipadika**, bahuvacana, ṣaṣṭhī, puṁliṅga, akārānta,* root(*rāma*)).
3. *su au jas am aut śas ṭā bhyām bhis ṅe bhyām bhyas ṅasi bhyām bhyas ṅas os ām ṅyos sup* 4.1.2
   pratyayaḥ 3.1.1
   paraśca 3.1.2
   We can see the importance of obtaining the attribute *prātipadika* to the nominal stem *rāma*. This encourages us to make a data structure that keeps on adding the attributes to a word for further usage in the rules of the *Aṣṭādhyāyī.*
   The application of this rule needs us to be familiar with the devices of *anuvṛtti* and *adhikāra* adopted by Pāṇini. The device of anuvṛtti aims at avoiding repetition of the same item. The device of adhikāra is used to indicate homogeneity of topic. The adhikāras stand for a subjectwise division of contents of the *Aṣṭādhyāyī.* The adhikāra *pratyayaḥ* 3.1.1 governs the rules in adhyāyas 3-5 and tells us that the items prescribed by these rules are called pratyaya. Further, the rule *paraśca* 3.1.2 -'That which is called a *pratyaya* is placed after the crude form', has its anuvṛtti till the end of chapter five. Thus, both these rules are applicable in the current rule and thus these affixes get the attribute of *pratyaya* and are applied after *rāma.*
   Form: *(rāma(prātipadika, bahuvacana, ṣaṣṭhī, puṁliṅga, akārānta,* root (*rāma*)) + **sup(pratyaya)**.
4. *tāni ekavacanadvivacanabahuvacanāny ekaśaḥ* 1.4.102
   supaḥ 1.4.103
   The array of 21 affixes will be transformed to a 7x3 array with the columns getting the attributes *ekavacana*, *dvivacana*, and *bahuvacana.* Each triad is called *vibhakti.* By matching *vibhakti* and *vacana*, we get the form:
   rāma(prātipadika, bahuvacana, ṣaṣṭhī, puṁliṅga, akārānta, root(*rāma*)) + **ām**(*sup*, **upadeśa, pratyaya, vibhakti, bahuvacana, ṣaṣṭhī,** root(*ām*)).
5. *yasmāt pratyayavidhis tadādi pratyaye'ṅgam* 1.4.13
   suptiṅantam padam 1.4.14
   After the application of these two rules, the structure is:
   (*rāma(prātipadika, bahuvacana, ṣaṣṭhī, puṁliṅga, akārānta*, **aṅga**, root (*rāma*)) + *ām(sup, upadeśa, pratyaya, vibhakti, bahuvacana, ṣaṣṭhī,* root (*ām*))) (*pada*).
6. *yaci bham*: *bha saṃjñā* is given to *rāma*. Form: (*rāma(prātipadika, bahuvacana, ṣaṣṭhī, puṁliṅgaakārānta, aṅga*, **bha**, root(*rāma*)) + *ām(sup, upadeśa, pratyaya, vibhakti, bahuvacana, ṣaṣṭhī,* root(*ām*))) (*pada*).

7. The above data structure is a nimitta of the following sūtras:
   ād guṇaḥ 6.1.87
   akaḥ savarṇe dīrghaḥ 6.1.101
   hrasvanadyāpaḥ nuṭ 7.1.54
   We run the conflict resolution module and the rule 7.1.54 is the winner rule. We have the insertion of *nuṭ* to the pratyaya *ām*. Thus, by the rule *ādyantau ṭakitau*, we have the following form (after passing through the *it* module:
   (*rāma(prātipadika, bahuvacana, ṣaṣṭhī, puṁliṅga, akārānta, aṅga, bha*, root (*rāma*)) + *nām(sup, upadeśa, pratyaya, vibhakti, bahuvacana, ṣaṣṭhī*, **āgama (nuṭ), ṭ-it, u-it**, root(*nām*))) (*pada*)
8. The above data structure is a nimitta of the following sūtras:
   nāmi 6.4.3
   supi ca 7.3.102
   After running the conflict resolution module, we get 6.4.3 as the winner rule. Thus, we have the form (after lengthening of the final *a* of *rāma*):
   (**rāmā***(prātipadika, bahuvacana, ṣaṣṭhī, puṁliṅga, akārānta, aṅga, bha* root (*rāma*)) + *nām(sup, upadeśa, pratyaya, vibhakti, bahuvacana, ṣaṣṭhī, āgama (nuṭ), ṭ-it, u-it* root(*nām*))) (*pada*)
9. No other rule from the *sapāda saptādhyāyī* is applicable and the structure moves to the *tripādī* after getting the attribute of *avasāna*.
   virāmaḥ avasānam 1.4.110
   (*rāmā(prātipadika, bahuvacana, ṣaṣṭhī, puṁliṅga, akārānta, aṅga, bha* root (*rāma*)) + *nām(sup, upadeśa, pratyaya, vibhakti, bahuvacana, ṣaṣṭhī, āgama (nuṭ), ṭ-it, u-it* root(*nām*))) (*pada*, **avasāna**)
10. *aṭkupvāṅnumvyavāye'pi* 8.4.2
   The rule changes the *n* of *rāma* + *nām* to *ṇ* and the structure is:
   rāmāṇām (pada, avasāna) *(rāmā(prātipadika, bahuvacana, ṣaṣṭhī, puṁliṅga, akārānta, aṅga, bha* root(*rāma*)) + *nām(sup, upadeśa, pratyaya, vibhakti, bahuvacana, ṣaṣṭhī, āgama(nuṭ), ṭ-it, u-it* root(*nām*)))

Formal Structure of Sanskrit Text: Requirements Analysis for a Mechanical Sanskrit Processor

Gérard Huet

INRIA Rocquencourt,
BP 105, 78153 Le Chesnay Cedex, France

Abstract. We discuss the mathematical structure of various levels of representation of Sanskrit text in order to guide the design of computer aids aiming at useful processing of the digitalised Sanskrit corpus. Two main levels are identified, respectively called the *linear* and *functional* level. The design space of these two levels is sketched, and the computational implications of the main design choices are discussed. Current solutions to the problems of mechanical segmentation, tagging, and parsing of Sanskrit text are briefly surveyed in this light. An analysis of the requirements of relevant linguistic resources is provided, in view of justifying standards allowing inter-operability of computer tools.

This paper does *not* attempt to provide definitive solutions to the representation of Sanskrit at the various levels. It should rather be considered as a survey of various choices, allowing an open discussion of such issues in a formally precise general framework.[1]

Keywords: Sanskrit, computational linguistics, finite-state machines, morphophonemics, dependency grammars, constraint satisfaction.

Introduction

We assume from the reader a minimal knowledge of the Sanskrit language and its grammar, some familiarity with general linguistics, and a general acquaintance with the use of computers in humanities. We shall evoke various formal structures used in current research in computational linguistics. Such material is of a mathematical nature, building on the last 40 years of research in mathematical logic (proof theory) and theoretical computer science (formal language theory, theory of computation, type theory, functional and constraint programming). Since familiarity with such formal methods is not assumed, and since their precise exposition would require more time than can be expected from most readers and more space than can be accommodated in a short paper, we shall skip technicalities and limit ourselves to rather sketchy discussions of their suitability for

[1] This material was presented at the Second International Symposium on Sanskrit Computational Linguistics, organized at Brown University in Providence by Peter Scharf in May 2008. Many thanks to Brendan Gillon, Amba Kulkarni and Peter Scharf for their remarks and corrections.

G. Huet, A. Kulkarni, and P. Scharf (Eds.): Sanskrit CL 2007/2008, LNAI 5402, pp. 162–199, 2009.

the representation and processing of Sanskrit text, while providing pointers to the literature for background reference.

The general architecture of a software platform aiming at linguistic processing usually consists of processing modules which operate at various levels of representation of the target corpus, typically across linguistic layers (phonetics, morphology, syntax, semantics, pragmatics), in order to progressively massage raw input (speech or written text) into higher level abstractions where enough meaning has been extracted for the task at hand (markup, intelligent search, concordance computation, statistics extraction, posibly up to more sophisticated cognitive tasks such as question answering and translation). Such modules communicate with each other in complex ways, using various databases of lexical and grammatical information compiled from structured digital lexicons. How many representation levels are identified is often a matter of hot debate. In particular there is a tension between theoretical abstractions from general linguistic models and pragmatic methods tailored to specific language families. We propose in this paper to identify two main levels for Sanskrit text structure, respectively called *linear* and *functional* level.

The first one, as its name suggests, concerns the *linear structure* of Sanskrit text: discourse consisting of successive utterances, sentences made up of streams of word forms, words themselves represented as strings of phonemes on which morpho-phonetic transformations operate, etc. At this level we find Sanskrit text represented as sequences of inflected words (*padapāṭha*), as sandhied continuous utterances (*saṃhitapāṭha*), or in a mixed manner as sequences of chunks facilitating the recognition of word boundaries while preserving the phonetic stream; at a finer grain of linear analysis we get marked-up text represented as sequences of morphemes tagged with morphological features. We shall discuss the feasibility of the synthesis of such descriptions by semi-automatic computer tools, and consider their suitability as a substratum layer of Sanskrit digital libraries fit for philological or pedagogical applications.

The next level concerns the *functional structure*, where various linguistic and logic formalisms for hierarchical structure (the main ones being phrase-structure syntax trees, dependency graphs, and proof nets for categorial grammars) are compared and their suitability for Sanskrit discussed. A formalisation of Gillon's notation for the syntactic analysis of Apte's examples is attempted, and alternatives are discussed. At this level we also find functional representations with thematic roles à la *kāraka*, encompassing Kiparsky's analysis of Pāṇini's grammar, and we propose to view those as enrichments of dependency structures. The articulation of the two levels, specially concerning the representations of compounds, is examined.

The computational implications of such design choices are discussed. Current solutions to the problems of mechanical segmentation, tagging, and parsing of Sanskrit text are briefly surveyed in this light. An analysis of the requirements of linguistic resources such as banks of lemmatized inflected forms or tree-banks used for statistical training is provided, in view of promoting the development of standards allowing inter-operability of computer tools.

1 The Linear Structure

We first discuss the representation of Sanskrit sentences as a (possibly annotated) list of contiguous chunks, down to words, morphemes, syllables and phonemes.

1.1 Phonemic Versus Syllabic Representations

First of all, we assume that the linguistic data is presented in a discretized fashion, that is we do not consider representations of continuous media such as phonetic waves, and we assume that the data is presented to us as streams of discrete characters from a finite alphabet. The main choice here is between streams of phonemes and streams of characters from some written representation. Roughly speaking, the first one is the phonemic representation, the second one the syllabic representation. The raw representation of a Sanskrit text is usually given as a syllabic representation, say in the Devanāgarī script. Digitalisation of this printed representation, by optical character recognition (OCR) or by direct keyboard input, produces strings of characters, some of which representing syllables, some of which representing punctuation symbols (*danda*, *avagraha*, spaces), possibly some extra characters such as numerals. These characters are represented in the computer in some low-level technology which is actually of little concern to us in this general discussion; the current technology uses Unicode values (code points) represented in some encoding scheme (such as UTF-8) as bytes of data (i.e. contiguous chunks of 8 bits in computer memory).

The set of phonemes of classical Sanskrit has been fixed since the seminal work of Pāṇini 24 centuries ago as a discrete collection of roughly 50 phonemes[2], which we shall simply represent as the first 50 natural numbers. These phonemes have been traditionally represented by Western sanskritists as romanized symbols with diacritics, such as *ā*, *ṭ*, *ś*, with a collating order inherited from ancient Sanskrit phonetic treatises, corresponds to the standard integer ordering of our integer codes. Actually, one small difficulty is often swept under the rug, concerning the nasalisation *ṃ* (*anusvāra*) and the final aspiration *ḥ* (*visarga*). We shall explain this point below. Finally, strings of romanized symbols are represented in the computer as transliterations, i.e. strings of ASCII characters. Popular transliteration notations are the Velthuis scheme, the Kyoto-Harvard scheme, the WX scheme used in Hyderabad and Tirupati, and the SL scheme used in the Sanskrit Library effort at Brown University. Appendix A gives the list of phonemes from classical Sanskrit under these various encodings. Many other encodings have been proposed in the past (ITRANS, CSX) but seem to become progressively obsolete.

The syllabic representation is not appropriate for basic computational linguistics tasks, and should be reserved to the Devanāgarī interface, since its granularity does not allow word segmentation of continued utterances, where syllables at word junction are often merged by sandhi. This problem is aggravated by the complex ligatures of the full Sanskrit Devanāgarī character set. Furthermore, one should

[2] In the broad sense of the term described by Scharf and Hyman [44] Section 6.1.6.

carefully distinguish between syllabic representation and Unicode representation of text. Unicode ought to be reserved for robust external exchanges, where one does not make explicit the encoding of ligatures; rather, successive consonants may be interspersed with *virāma* codes, and it is the rendition engine of the printing host which decides whether fonts available on this host may support a given ligature or not. Actually, Unicode points may be used for representing phonemes, syllables, printable letters, or control characters, and the ensuing ambiguity makes this low-level medium unfit for precise linguistic computation. Even if one chooses to represent the syllabic level, for prosody purposes for instance, a specific precise representation separate from Unicode ought to be used.

From now on, we shall assume that the basic text representation is phonemic. Thus raw output in syllabic representation has to be expanded in the finer phonemic representation. This looks straightforward enough, since every syllable is non-ambiguously translatable into a string of phonemes. However, difficulties occur when one wants to use spacing, or special symbols such as avagraha, in order to disambiguate sandhi and have a representation which makes explicit the word boundaries. Thus raw input is generally broken into *phonemic chunks* separated by blank space, each chunk formed of Sanskrit words merged with sandhi, so that a given word (*pada*) is entirely contained within one chunk. We shall discuss in the next section various representations of Sanskrit text as lists of phonemic sentences, each phonemic sentence being represented as a list of phonemic chunks.

1.2 Sandhi

We have been a little vague in referring to the process of *sandhi*, i.e. transformation of contiguous phonemic strings. Actually, there are at least two rather distinct such processes, traditionally called respectively *external* and *internal* sandhi by Western linguists. Basically, external sandhi is the transformation that occurs at word boundaries, and when merging a raw nominal stem (*prātipadika*) with an inflected form in order to compose a compound form. Whereas internal sandhi occurs between morphemes at a deeper level of morphological formation, either for derivational morphology (*kṛt* formation of nouns and participles from roots, *taddhita* secondary word formation), or for inflectional morphology (declension of substantival and adjectival stems, given gender case and number attributes, and verb conjugation given various morphological parameters). Actually, whereas external sandhi is more or less unique, internal sandhi is more a family of transformations under various circumstances. Indian grammarians do not use this terminology, which does not correspond to a precise Pāṇinian notion. Roughly speaking, the different varieties of internal sandhi correspond to various situations where a context of previous transformations determines the possible application of further rules. However, it is not usually directly possible to map a given morphological process (say conjugation) into a unique entry point in Pāṇini's *sūtras*. Furthermore, Pāṇini's *economy principle* forces the sharing of many transformation rules, which may be used for external sandhi in certain contexts, or to varieties of internal sandhi in others, making the distinction non relevant to Pāṇinian processes.

We believe nonetheless that the distinction between internal and external sandhi is conceptually important, since their computational properties are very different. First of all, in terms of modern formal language theory, external sandhi can be defined as a *rational* transduction between phonemic strings, expressible as a regular expression. In other words, external sandhi, and its reverse operation sandhi splitting, is a local transformation which may be implemented by a finite machine (more precisely, a finite-state transducer) operating on strings of phonemes. This is not the case for internal sandhi, which has long-distance effects such as retroflexion, not easily amenable to finite state techniques. Furthermore, internal sandhi does not really operate simply on phonemic strings such as stems, but also on morphological affixes which use intermediate control characters before their erasure in the terminal surface representations. The fact that Pāṇini used actual phonemes to encode such meta-theoretic control characters (*pratyāhāra* markers, etc.) is an artifact of his system, where meta notation is immerged within the object language (the *signifiants* or phonetic realisations). There is no actual necessity to use these encodings, and it may be better to have a clearer distinction between morphologico-phonetic markers (meta-linguistic symbols) and actual phonemes.

The next level of representation of Sanskrit text is as a list of *words*, possibly annotated by the (external) sandhi rules applicable at their junction. Here by word we mean either an indeclinable form (some adverbs, prepositions, conjunctions and other particles), a declined noun stem, or a conjugated verbal form. Actually, this representation is too coarse, since it is not finite, in the sense of being finitely generated from a fixed set of primitive forms, since compounds may compose individual stems (*prātipadika*) to an arbitrary level. In order to obtain a finitely based representation, one has to break up compounds. Declined compound forms may be split as a stem form followed by a shorter declined form, which may be itself split down to a non-compound form, of which there is a finite vocabulary. Similarly an indeclinable compound (*avyayībhāva*) may be split into an adverbial prefix and a stem form. Such splitting of compounds may be done mechanically by the same process of (external) sandhi analysis (*viccheda*) that may be used to break chunks of sentences into words.

By the same token, one may decompose verbal forms into a sequence of preverb particles, followed by a root form, although in certain cases a more complex glueing may be involved, with possible retroflexion, such as *adhiṣṭhā* from preverb *adhi* and root *sthā*. Furthermore, one cannot hope to obtain such effects through just sandhi rules, which operate on contiguous morphemes, as remarked by Kiparsky in [29]. For instance, consider the perfect form *abhitaṣṭhau*: there is no hope to effect the retroflexion induced by prefixing *abhi* to the root *sthā* by a local interaction of *abhi* with the reduplicated passive form *tasthau*. Thus the interaction has to be predicted by storing forms such as *taṣṭhau*, accessible only through the relevant preverbs. Similarly, special forms have to be cooked for the preverb *ā*- interacting with forms of roots starting with *i, ī, u,ū*, in order to segment properly say *ihehi* as *iha-ā-ihi*, while rejecting the wrong **ihaihi*.

There is no space to describe these mechanisms in detail, we just remark that modular finite-state methods [28] allow to encode such controled interactions in state transitions rather than in the conflict resolution of complex morphophonemics rules.

1.3 Forms of Surface Representation

At this level of analysis of a Sanskrit sentence, we obtain a surface representation which is linear in the sense of being a uni-dimensional list of items from a finite list of atomic forms. Let us give an example, taken from [29]. We give various forms of linear representation.

devanāgarī:
वनाद्ग्राममद्योपेत्यौदन आश्वपतेनापाचि
romanized continuous text (*saṃhitāpāṭha*):
vanādgrāmamadyopetyaudana_āśvapatenāpāci
saṃhitāpāṭha (in VH transliteration):
`vanaadgraamamadyopetyaudana_aa"svapatenaapaaci`
list of word forms in terminal sandhi (*padapāṭha*):
`vanaat graamam adya upetya odana.h aa"svapatena apaaci`
Chunk form (identical to the one given in [29]):
`vanaad graamam adyopetyaudana aa"svapatenaapaaci`
Surface representation with explicit sandhi annotation:
vanaat⟨t|g→dg⟩graamam⟨⟩adya⟨a|u→o⟩upetya⟨a|o→au⟩odana.h
⟨a.h|aa→a_aa⟩aa"svapatena⟨a|a→aa⟩apaaci
One may check that this surface representation is exactly the padapāṭha with spaces replaced by explicit sandhi annotation, and that the saṃhitāpāṭha is obtained from it by effecting the sandhi rewriting.
Fully segmented form (breaking down preverbs from root forms):
vanaat⟨t|g→dg⟩graamam⟨⟩adya⟨a|u→o⟩upa⟨a|i→e⟩itya⟨a|o→au⟩odana.h
⟨a.h|aa→a_aa⟩aa"svapatena⟨a|a→aa⟩apaaci

Another possible one:
vanaat⟨t|g→dg⟩graamam⟨⟩adya⟨a|u→o⟩upa⟨a|aa→aa⟩aa⟨aa|i→e⟩itya⟨a|o→au⟩
odana.h⟨a.h|aa→a_aa⟩aa"svapatena⟨a|a→aa⟩apaaci

The two fully segmented forms differ only by the preverb of the absolutive form *upetya*, obtainable as *upa-itya* or *upa-ā-itya*. Remark that in this distinction the verb *upe* is ambiguous in all forms. This example is interesting in several ways. First of all, it points to the fact that preverbs are glued by sandhi one at a time, left to right; it would be wrong to affix the preverb *upa* to the absolutive *etya* of verb *e* = *ā-i*, since this would lead to the wrong form **upaitya*.[3] This has as consequence that forms of *upe*=*upa-ā-i* are undistinguishable from corresponding forms of *upe*=*upa-i*, at least at this level of phonemic distinction. This

[3] Unless a special rule is formulated to account for the exception as Pāṇini did in 6.1.95 *omāṅgoś ca* (P. Scharf).

does not mean that the two verbs ought to be considered the same, since they have different shades of semantic meanings; consider *ihehi* (come here), obtained as *(iha-ā)-ihi*, and not as *iha-ihi*. Indeed Monier-Williams gives two distinct entries for *upe*. And this distinction justifies our explicit consideration of the fully segmented form, since the surface representation (or padapāṭha) does not distinguish the two.[4] In this particular example, it is the last form which represents faithfully the meaning, according to the analysis in Kiparsky [29].

1.4 Splitting Compounds

The fully segmented form ought to make explicit the decomposition of compounds as well, in order to satisfy our requirement that it be finitely generable (from a finite lexicon of basic stems). Thus a compound form such as *tapovanavasitasukhā* ought to be broken up as *tapas-vana-vasita-sukhā*, possibly with sandhi-annotation between the components.

Several remarks are in order. First of all, note that the semantic understanding of such a compound demands a more complex representation than just a left-to-right list of stems. In general, a binary tree indicating the dependency ordering is needed,[5] and the more standard compounds get their main constituent to the right, such as *(((tapas<vana)<vasita)<sukhā)* in Gillon's notation. Whereas a simple list such as *tapas-vana-vasita-sukhā*, parsed from left to right, has a derivation tree going in the other direction: *(tapas-(vana-(vasita-sukhā)))*, and its sublists are not proper constituents. We should not be surprised at this. The sandhi splitting of compounds does not reflect its semantics or even its syntax, just the morphology rule stating inductively that a compound form is obtained by prefixing a bare non-compound stem to an inflected form (possibly itself compound). A more general rule, closer to the semantics, would express that the left component of a compound is a possibly compound stem (*prātipadika*). But this more general rule expresses a non-regular construction requiring non-linear representation (i.e. context-free derivation trees), and we do not want to consider it at this stage. In other words, we only keep of compound morphology what pertains to its finite-state character, and leave the rest to the non-linear structure, which accommodates syntax and possibly semantics, and which will support a better structured representation of compounds. It is exactly at this point that we may state a clear requirement of the interface between the linear and the functional levels: the linear structure of compounds ought to be an embedding

[4] While it is common for editors of classical Sanskrit texts not to separate preverbs from verbs even when indicating other word divisions, and for some lexicographers, such as Monier-Williams and Apte, to list the preverb-verb unit as a headword, Vedic padapāṭhas do in fact separate preverbs and verbs as independent words. For example, RV 1.124.8 saṃhitāpāṭha *ápaity* is analyzed *ápa — eti —* in the padapāṭha. The nominal form *prétim* in RV 1.33.4 is analyzed — *prá'itim* — separating the preverb with an avagraha indicated in Roman by an apostrophe (P. Scharf).

[5] Pāṇinian compound formation rules (2.2.23-24 *śeṣo bahuvrīhiḥ*, *anekam anyapadārthe*, and 2.2.29 *cārthe dvandvaḥ*) allow the derivation of bahuvrīhi and dvandva compounds from multiple constituents (P. Scharf).

of its functional representation. This is accomplished simply by requiring that the list of stems be the *frontier* of the binary tree. For instance, in the above example, the list *(tapas-(vana-(vasita-sukhā)))* of phoneme strings is the frontier of the binary tree *(((tapas<vana)<vasita)<sukhā)*. More generally, this will simply state that the frontier of the functional representation should be the sequence of utterances, the phonemic realisation. In other words, we are requiring the functional representation to be sequentialised on the enunciation – no movement, no erasure, no duplication. This requirement is reasonable, it amounts to requiring that the functional representation be a mark-up of the enunciation.

Another remark is that it may be undesirable to split compounds which are used as proper names, such as *devadatta*, or *rāmāyaṇa*, since their meaning is in general non-compositional. However, an automated tagger will not be able to recognize such proper names, since they are not distinguished by capitals or other marks. At best it might try not to split a compound explicitly presented in the lexicon as a proper name, but even then it will not be able to distinguish a genuine compound *devadatta* "god-given" from the proper name *Devadatta* "Theodore", anymore than it would be able to discern the color black from the name *Kṛṣṇa*. We note with interest the use of capitals as initials of proper names in Gillon's presentations or in the Clay Sanskrit Library series of books. We have adapted the same convention in our lexicon, and for this reason use for transliteration a reduced set from Velthuis'; e.g. *A* is not allowed as substitute for *aa*.

Finally, it is not clear whether privative compounds (such as *a-vyakta*), and compounds whose left component is a particle such as *vi-*, *sa-*, *su-* or a preposition ought to be split, even though their meaning is in general compositional. Such compounds may be considered under derivational morphology rather than compound formation proper, and their analysis left to lexicon lookup rather than sandhi-splitting. This is debatable of course, and ultimately too naive, since e.g. privative *a-* occurs as initial of arbitrarily long compounds. Similarly there is a trade-off between storing explicit entries in *-tā*, *-tva*, *-tṛ*, *-vat*, *-mat*, etc. in the lexicon, versus leaving them implicit from derivational morphology considered as generative. We remark *en passant* that finite state automata allow to represent implicitly an infinite number of forms, in contrast with say relational database technology, where the stored data is necessarily finite.

1.5 Automated Synthesis of Surface Representation

This discussion calls for closer examination of the various mappings between the representations discussed above, in order to understand how to build mechanical tools that may transform one into another.

We already remarked that the saṃhitapāṭha may be obtained from the padapāṭha by applying (external) sandhi.[6] This transformation is deterministic (i.e. may be implemented by a function), except minor details such as doubling of *n* before vowels or the doubling phenomena described in [33]. It becomes fully

[6] We ignore exceptions, such as presented by Staal [46] p. 22, where the padapāṭha differs in the word order of the saṃhitapāṭha in a verse of the *Ṛgveda*, in order to restore the proper name *śunaḥśepa* split by particle *cid*.

deterministic of course if we annotate the padapāṭha with the relevant sandhi rules.

In the reverse direction, we cannot hope for a functional transformation, but since external sandhi is a rational relation [5], its inverse (*viccheda*) is too. Furthermore there is a finite number of strings which are the viccheda of a given saṃhitapāṭha, and thus all solutions may be enumerated by a terminating finite state process. Details are given in [25]. This algorithm is complete, assuming that all forms occurring in the sentence are in the database of inflected forms, i.e. that the lexicon used for morphology generation contains the vocabulary of the analysed sentence.

For instance, in the Sanskrit engine platform, I marked the stem *nṛ* as non-generative, since its nominative form *nā* was overgenerating wildly in the segmentation phase, every *nā* syllable having its chance as a nominal segment. Thus I cannot properly segment the sentence *ekonā viṃśatirnāryaḥ krīḍāṃ kartuṃ vane gatāḥ* 'one man and 20 women went to play in the woods'. But in some sense this example vindicates the decision, since this sentence is very likely to be misunderstood, being a sort of garden path sentence, as the linguists say – you are fooled by the likeness with *ekonaviṃśati* to believe that only 19 women went to the woods, and you are then puzzled by the second half of the *prahelikā*: *viṃśatirgṛhamāyātāḥ śeṣo vyāghreṇa bhakṣitaḥ* '20 came back, and the remaining one was eaten by the tiger'. The moral of this *vakrokti* is that *nā* is problematic, even for human locutors[7] since its frequency as a syllable so much exceeds its frequency as a form that it is unreasonable to handle it with the standard combinatorics, and it must be treated as an exception somehow. A similar problem occurs with *āya*, whose declined forms clash with genitive/ablative forms of substantives in *ā* and dative forms of substantives in *a*, and thus cause overgeneration. Such examples raise the general question of ambiguity between generative and inflexional morphology.

Actually, this viccheda algorithm is not usable in practice as a stand-alone segmenter without refinements, since the number of segmentations may be exponential in the length of the sentence, and thus the problem is computationally intractable for large continuous chunks of saṃhitapāṭha. Thus for the example sentence discussed in the section 1.3, with no blank at all in the input, the number of possible segmentations exceeds 10000. The standard solution is to break the input at appropriate points in order to give hints. Of course if one goes all the way to preparing by hand the padapāṭha the problem becomes trivial, involving only an iteration of lookups in the database, and the only remaining non-determinism reduces to proper homonymy.

Somewhere in between, we may obtain a useful algorithm, by breaking the input with spaces at a few chosen points, like in the "chunk form" considered above. A remark is in order here. The chunk form is definitely *not* the padapāṭha, and indeed the two forms are mutually inconsistent. For instance, a chunk ended by *o* will generally correspond to a terminal sandhi form *aḥ*, as opposed to being

[7] Admittedly, the machine would not be fooled on that particular example, because of the discrepancy between *a* and *ā*.

the terminal sandhi form in *o* of e.g. the vocative of an *u* stem. Also a chunk ended in *a* in a hiatus situation will generally correspond to a terminal form in *aḥ*, *e* or *o*. Finally an avagraha may be used to signal the elision of an initial *a* after *e* or *o*.

Actually, there does not seem to exist a commonly agreed chunk representation, specially concerning the use of the avagraha. We have evolved an ad-hoc convention for chunk representation, which attempts to depart from the saṃhitapāṭha only in inserting blanks at points which must be word junctions. This seems pretty close to current digitalised Sanskrit texts, and is consistent with our sandhi splitting algorithm. The precise algorithm is given in Appendix B, and may be read as a specification of the chunk representation preprocessing.

This algorithm accepts input such as *yad iha asti tad anyatra yan neha asti na tat kvacit*, in order to pass it on as a list of final forms, to the sandhi viccheda segmenter proper. As analysed in [25], this problem is solvable by a finite-state transducer, implementing the inverse of the sandhi relation. Furthermore, for any input, there is a finite set of solutions. For instance, the transducer on the above input will infer the correct padapāṭha segmentation *yat iha asti tat anyatra yat na iha asti na tat kvacit*. However, 19 other segmentations are possible, since e.g. *neha* is a form of root *nah* in the perfect tense, and admits a total of 5 decompositions. Similarly *kvacit* admits 4 decompositions, amazingly enough. Since the combinations multiply, we get 20 decompositions. At this point the reader may wonder whether such a parser is usable in practice. The answer is that in this initial phase it is only a segmenter, piping into a semantic filter, which then attempts to build a satisfiable functional representation of each segmentation, using a kāraka-fitness test. In this precise example, this filter rejects all 20 segmentations except two, the good one and a parasitic one. Thus there is no silence, and the noise is tolerable: the precision is 94% in this case. It is rather surprising that on such an input, where most word divisions are explicitly indicated, there is still room for 20 solutions. It is interesting that if all spaces are removed, and we give the input in full saṃhitapāṭha as *yadihāstitadanyatrayannehāstinatatkvacit*, there are now 87 potential segmentations. In this case, the filter keeps 9, the good one and 8 parasitic, with precision 91%. This gives a measure of the tradeoff between undoing sandhi by hand and leaving it to the machine. If you give proper hints, a few spaces in a moderately long sentence, you get a manageable set of proposed solutions to inspect in order to select the right parse.

The example sentence above: *vanād grāmam adyopetyaudana āśvapatenāpāci* currently yields 928 segmentations, among which only 2 are kept, the right one being 1st. The other one is due to the semantic ambiguity of *upetya* (with or without preverb *ā*).

1.6 Normalisation

It may be appropriate at this point to discuss the status of anusvāra coming from internal sandhi. Typically, the word *sandhi* itself comes from (obligatorily) compounding the preverb *sam* with the *kṛdanta* form *dhi* derived from the root *dhā*. And indeed the orthography *saṃdhi* is perfectly legitimate, anusvāra in

this case being an alternative to nasal homo-phonic to *dh*, i.e. *n*.[8] Are we going to need to store redundant forms *saṃdhi* and *sandhi* in our lexicon, in order to store duplicate forms of all their inflected forms? In order to recognize them in all possible combinations in the input, yielding a potential exponential number of completely redundant segmentations? Of course not. We need store only one canonical form of the stem, *sandhi*[9] one canonical form of each of its inflected forms such as *sandhiḥ*, and have chunks *saṃdhiḥ* in the input normalized in their canonical form *sandhiḥ* at preprocessing time. Note that this is the right normalisation direction, since it would be incorrect, using the reverse convention, to write e.g. *sasañja* as **sasaṃja*.[10] Of course "genuine" anusvāra remains in e.g. *saṃsāra*. Symetrically, we propose to normalize *ahaḥsu* as the equivalent *ahassu*.

This normalisation of the input is one of the tricky technical details. Note that it implies a reconsideration of sandhi rules, since for instance one choice of sandhi of *sam* and *diṣṭa* produces the un-normalised *saṃdiṣṭa*. Thus there is a dilemma here. Either the lexicon has an explicit entry for the participle, normalised as *sandiṣṭa*, or the lexicon attempts to generate this form as composed of preverb *sam* and root participle *diṣṭa*, in which case the sandhi rule to be used should be *m*+*d* →*nd* rather than *m*+*d* →*ṃd*. This technical issue is tricky, since we do not want our morphological generator to overgenerate by allowing variations inconsistent with the normalisation of the input. Thus by the economy principle we prefer to interpret sūtra 8.4.59 by "is preferably replaced" rather than "is optionally replaced".

More generally, the treatment of participles is a thorny issue. There seems to be no good reason not to restrict the explicitly represented participles to the roots, and to get the other ones by preverb affixing, in the same way that we get finite verbal forms from finite root forms. But here two issues come into play. One, there are so many participial forms, and their full inclusion leads to many ambiguities. A typical one is the ambiguity between a present participle in the locative singular such as *bhavati* and the 3rd person singular of the corresponding present conjugation. Secondly, generating participial forms from roots is a special case of the first level of generative nominal morphology (*kṛdanta*), and thus it would seem logical to include more, or all *kṛdanta*s, or even *taddhita*s, and this looks like opening Pandora's box, with overgeneration by unattested forms on one hand, and complex internal sandhi inversion problems on the other.

[8] The phonetic alternation of *n* and *ṃ* is due to the fact that pada-final anusvāra is optionally replaced by the nasal homorganic with the following stop or semivowel in accordance with 8.4.59 *vā padāntasya*. The preverb *sam* is a pada which, as an upapada, is compounded obligatorily with the kṛdanta *dhi* in accordance with 2.2.19 *upapadam atiṅ* (P. Scharf).

[9] Note that Monier-Williams takes the opposite choice, and chooses *saṃdhi* for the corresponding entry of his dictionary.

[10] Non-pada-final anusvāra is obligatorily replaced by the nasal homorganic with the following stop or semivowel in accordance with 8.4.58 *anusvārasya yayi parasavarṇasya* (P. Scharf).

1.7 Lemmatisation

We are not done yet with the linear representation, since now we may invert morphology on the individual elementary forms. Inverting morphology is not hard, if forward morphology has been already implemented, since the lexicon may be compiled into all possible forms, and this database of basic forms be stored as persistent dictionaries where they can be looked up by a stemmer. This assumes the existence of lexicons having sufficient coverage of the vocabulary of the target corpus. This requirement is not so severe, in view of the fact that our phrasal segmenter segments compounds into their constituents, so recursively only root words have to be lexicalized, besides a finite collection of irregular compounds. Besides, there exist now XML versions of the Monier-Williams dictionary,[11] and this is certainly enough to cover a significant part of the corpus. Actually, it may be too large for the task. The problem is the generation of verbal forms from such a huge dictionary. Such a dictionary covers well the nominal forms, but verbal forms have to be generated; all that is explicit are a few representative forms of a few tenses. Worse, most participles are missing, only past passive and a few passive future participles are explicitly listed. The various forms of the present participles, active past participles, and perfect participles are usually omitted as implicitly understood.

Thus forward morphology has to be implemented, not only for nouns, which is easy, but for verbs, a much harder task. This process may be more or less Pāṇinian, so to speak. For instance, in my own system, I more or less follow Pāṇini for internal sandhi, but generate paradigm tables for computing inflection in a rather ad-hoc manner percolated from mostly Western grammar descriptions. What I mean by Pāṇinian for internal sandhi is that I go through operations which mimick Pāṇini's sūtras, but using some definite choice of deterministic computation by specific rules in a sequential manner. In doing that, I allow myself the freedom of deciding which rule is *siddha*, but then everyone is in the same situation, isn't it? You have to interpret the vārttikās, decide on the proper scoping of exceptions, interpret priority of rules by *siddha* or other considerations, etc. So for me being Pāṇinian is more a reference to the spirit of a genious computational linguist precursor. And I allow myself to abstract from the straightjacket of an encoding of computation in terms of conditional string rewriting systems. We have general purpose programming languages, with appropriate data structures, we know how to separate clearly object data (phonemes) from meta-notation (Sanskrit phonemes used as control codes such as the markers of *pratyāhārās*).[12] Should Pāṇini be reborn as a computational linguist of our times, he would use modern tools and methodologies, of course, frame his

[11] Such as the one resulting from the collaboration of the Cologne Digital Sanskrit Lexicon project and the Digital Sanskrit Library Integration project at Brown University.

[12] Abbreviatory terms formed by taking the first element of a list together with a marker placed at the end of the list. See the article by Amba Kulkarni in this volume (P. Scharf).

description of the language in terms of modular high-level processes, and leave the production of compact machine code to an optimizing compiler.

One decision which may be considered cavalier with Pāṇini was to split in two the phoneme *j* in order to make internal sandhi deterministic in view of the different behaviours of say *mṛj* and *bhuj* in *kta*-suffixes, cf. their past participles respectively *mṛṣṭa* and *bhukta*. Similarly *h* has three variants, one for *duh*-like roots, one for *lih*-like roots and one for *nah*-like roots, (compare participles *dugdha*, *līḍha* and *naddha*).[13] This story is told in [27]. Being *mṛj*-like or *duh*-like is lexicalized.

The method sketched above is lexicon-based. It will fail to lemmatize forms which are not generated from the morphological closure of the root entries of the digitalized lexicon. An easy analysis may account for all irregular compounds which are explicitly listed in the lexicon. Since these irregularities are hopefully non-productive, the lexicon may be made complete since there are a finite number of attested ones. Regular compounds generable from the lexicon, at any depth, will be taken care of by the segmenter, as explained above. But there will be silence for any other form.

There exist heuristic methods to infer root stems from forms guessed as inflexions of potential words. These heuristics may be tuned by optimization on a reference tagged corpus; this is statistical training. Such heuristics allow one to lemmatize a text automatically without the need for a complete lexicon, and indeed may be used to provide suggestions for lexicon acquisition. Such methods have been used successfully by [37]. However, one should understand that Sanskrit morphology is too rich for this method to be complete, since there is a heavy non-determinism in inverting internal sandhi; for instance, should a retroflex *ṇ* or *ṣ* be reverted to *n* or *s* is not an obvious decision, and it is not clear that statistical training is sufficient in the absence of large-scale reference corpora. Furthermore, this method works pretty well when the text is presented in padapāṭha form, but it is not clear that it extends easily to chunks. In particular, external sandhi would have to be undone, but this time in a right-to-left direction, while the method given in [25] works left-to-right, on finite transducers which are compiled from the lexicon of inflected forms, so a different technology would have to be used.

1.8 Tagging

Once lemmatization is understood, we may proceed to the final stage of our linear representation, the mark-up of the text by its morphology: each inflected form is marked with its lemmatization.

Let us give as example what our own tagger returns on the example:

```
vanaad graamam adyopetyaudana aa"svapatenaapaaci
```

```
[ vanaat        { abl. sg. n. }[vana]     <>]
[ graamam       { acc. sg. m. }[graama]   <>]
```

[13] Since *h* derives historically from any of the voiced aspirated stops in Indo-Iranian (P. Scharf).

```
[ adya          { adv. }[adya]             <a|u -> o>]
[ upaa - itya  { abs. }[upa-aa-i]        <a|o -> au>]
[ odana.h       { nom. sg. m. }[odana]    <.h|aa -> _aa>]
[ aa"svapatena { i. sg. m. }[aa"svapata] <a|a -> aa>]
[ apaaci        { aor. [1] ps. sg. 3 }[pac]
```

A few remarks are in order. The notation `<a|o -> au>` indicates sandhi. It may be omitted. The tags for substantival forms like *vanāt* are pretty obvious. Such a tag has an abstract form, such as `<Subst case=5 number=1; gender=3; stem="vana">`, we pretty-print here an ad-hoc sugared concrete form with braces. Braces are convenient; they may be used as a set notation for *multitags*. The above example is a lucky one, because it happens that there is only one morphological production of each form. In general, ambiguities in morphology will produce multitags, like for the form *vanebhyaḥ*, which lemmatizes as `{ abl. pl. n. | dat. pl. n. }[vana]`. Actually there is even a second level of non-determinism, if we want to discriminate between homonyms, but I shall not discuss this, since it presupposes that the approach is lexicon-based.

The tag for the finite verbal form *apāci* is similarly non-mysterious, the numeral 1 standing for the aorist formation type (from 1 to 7). Similarly, the present system formations are indexed by their *gaṇa* (from 1 to 10), and passive future participal forms are similarly indexed by 1, 2 or 3, according to their formation suffix *-ya*, *-anīya*, or *-tavya*.

This extra information is important to relate modern linguistic abstractions, reflected in the tags' algebra, and actual entry points in Pāṇini's grammar. Of course a purely Pāṇinian system will replace the tags altogether by their full simulation path in the grammar. This must be understood as the problem of interface of the inflected forms with the grammar component, and with proper design a standard set of tags ought to allow interoperability between a Pāṇinian generator and a paradigm-driven one, there is no deep incompatibility between the two approaches.

A complete discussion of verbal tags is a complex matter, and their standardization will need extensive debate. For instance, passive forms may be classified as forms of the root in passive voice (in a system with 3 recognized voices, active/*parasmaipada*, middle/*ātmanepada* and passive), the option currently chosen in my system, consistent presumably with a Pāṇinian treatment, as opposed to Whitney's treatment of passive as a secondary conjugation. The granularity of the morphological notation is also questionable. For instance, we refine the aorist tag by indicating its mode of formation (here 1 among the 7 forms of aorist), since this information is lexicalized; not all roots admit the 7 forms, so only listed modes are generated. Similarly, forms of the present system are tagged by the family class [*gaṇa*]. This is important for verbs which do not have the same regime in the various classes; e.g. the root *pac* is transitive in class 1 (*pacati*, *pacate*) and intransitive in class 4 (*pacyate*), and this information is crucial for the kāraka analysis which we shall discuss in Part 2 below.

The remaining tags concern the indeclinable forms *adya* and *upetya*. Here we make the choice of classifying such indeclinable forms according to their "part of speech" category, like `adv.` for the adverb *adya* or `conj` for the particle *ca*. The exact granularity of such categories is of course open to discussion. The form *upetya* is recognized as the absolutive of the verb *upe*, obtained from the root *i* by the sequence of preverbs *upa* and *ā*. This segmentation is effected by the segmenter, so that only forms of the roots have to be generated, and this is true as well of finite forms. Actually this example is a tricky one. Note that sandhi of preverbs *upa* and *ā*, leading to the form *upā*, has to be effected before the sandhi to the absolutive form *itya*. Otherwise, glueing the preverbs one by one in the etymological order would lead first to *etya*, then to the wrong **aitya*. This is a tough nut to crack for the segmenter, in view of the fact that the preverb *ā* is reduced to a single phoneme, allowing overlaps of left sandhi and right sandhi. This problem is solved in my implementation by a technical device called *phantom phonemes*, which is described in [26, 23]. More such technical devices are needed to account for the retroflection in forms such as *nirṇayati*, *viṣṭabdha*, etc.

A technical problem concerning absolutives is that the segmenter has to know that e.g. *-ya* is the form to be used with preverbs, while *-tvā* is the form to be used for plain root forms. This distinction is achieved by putting the two forms in distinct lexical categories (respectively accessed through the states Abso and Unde in the figure below), and having the segmenter operate transitions between such categories controled by a finite automaton whose states denote the phases of recognition of a simple regular grammar expressing preverb affixing, compound formation, periphrastic phrases and more generally morphological geometry. This mechanism is easily achieved via the technology of modular transducers [28]. Fig. 1 shows the current finite automaton used in my system.

A final remark is that perhaps this layer ought to be called the *rational* structure rather than the *linear* one, since its structure is not just a list of items, but a list of regularly structured items – reflecting the fact that compound formation, preverb affixing, and absolutive selection are definable as regular expressions, and solved by finite-state automata. This layer is the realm of finite-state methods, as opposed to the next layer, which represents an algebraic structure in the realm of context-free or even stronger combinatorial processes.

2 The Functional Structure

In the tradition of Western linguistics, specially when dealing with languages with strict word order, it is customary to distinguish a layer of syntax prior[14] to a level of semantics. The combinatorial analysis of syntax, in a perspective of general linguistics which applies with more or less felicity to free-order languages, has burgeoned into a immense mass of material since the middle of the 60's, when Chomsky started his program of formal syntax. It may be worthwhile to give a short overview of these developments.

[14] In the analysis direction from utterance to sense.

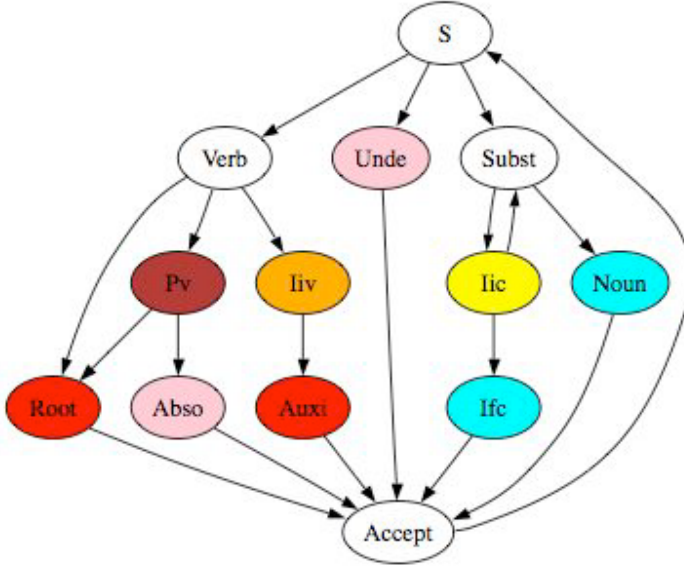


Fig. 1. The 9-phases lexical analyser

2.1 Short Survey of Formal Linguistics

The first step in giving more structure to the sentence than the sequential order of its utterance is to give some substructure hierarchy – another partial ordering – which reflects some well-parenthesized notion of *constituent*. Thus tree-like structures are proposed as syntactic notations, subtrees being phrases, terminal nodes being the words themselves. Typically, such phrase-structure trees may represent the derivation trees of some context-free grammar. Non-terminal symbols from the grammar label the structure as mark-up tags for the phrases sorted by these non-terminals – sentences, verbal phrases, nominal phrases, etc. More abstractly, some of the recursion structure may be omitted, so that derivation trees are abstracted into terms from some abstract syntax algebra. The two levels correspond to what the logician Curry called the phenogrammatics and the tectogrammatics, to what the linguists call respectively *shallow* and *deep* structure, or the computer science distinction between *concrete syntax* and *abstract syntax* (see the discussion in [4]). Abstract syntax is closer to the semantics of the linguistic elements, since it abstracts away from operational details about parsing and word order, and keeps the essential domination ordering. In the Indian grammatical tradition, the distinction *abstract* vs *concrete* syntax corresponds more or less to the distinction *saṃbandha/abhisaṃbandha* (see [46] p. 13).

The problem of relating the syntax and the semantics is that usually syntax permits dislocations of the sentence, allowing arguments of a given semantic operator to appear at possibly long distance in the precedence ordering of its utterance. In order to represent the derivation of the semantic structure from the syntactic one, as some kind of compilation process, linguists have

studied derivation sequences proceeding by movement of constituents, leading to so called *transformational grammars*. One proceeds from one structure to the next by some kind of computation step. This computation may be expressed by some outside process, like an interpreter, viewing each structural description as a data item. This is typical of rewriting formalisms, where each structure is like a term in some first-order logic, and computation is algebraic equality substitution in the relevant algebra. Alternatively, the structures may be represented in some higher-order formalism such as λ-terms, and the rewriting calculus is now the step by step unfolding of internal computation by λ-reduction, possibly guided by some strategy. Now non-terminals are no more atomic categories, like in phrase-structure grammars, but rather structured entities called *types*, similar to logical propositions. This last approach is characteristic of categorial grammars à la Lambek, where syntactic structures are functional expressions typed by an algebra of syntactic categories, and compile into semantic combinators in some typed λ-calculus, in the Montague tradition. Logical semantics is thus expressed as the normal form of the structure, and the combinators express the necessary movements, the precise order of evaluation, as well as the proper scoping management of quantifiers. Nowadays transformational grammars have somewhat fallen into disuse, after decades of development of complex ad-hoc devices, in favor of more uniform computation processes well studied from logic and fundamental informatics such as λ-calculus and linear logic.

The status of morphology in these various views about syntax reduces generally to annotation of the syntactico-semantic structures by morphological features. Those are used mostly for controling the constructions by consistency constraints such as agreement/concord or coordination, although some of them survive in the final semantics (typically number). These features may be organised with polarity distinctions (distinguishing between the producer of a feature, such as the case of a noun phrase, and its consumer, typically a verbal phrase which needs an argument or a complement of that case by its subcategorisation regime). These features may then be made essential components of the computation process; this is the view currently held by computational versions of minimalism such as Edward Stabler's.

It is sometimes hard to dissociate general algebraico-logical frameworks from linguistic theories, such as Chomsky's $\overline{X}$ theory. And it is not always clear whether a formalism is proposed as expressive enough for modeling the right notions, or whether it is crafted in such as way that it leads to tractable parsing algorithms. Let us recall that general context-free grammars are parsable in a time proportional to the cube of the length of the candidate sentence, and that more expressive mildly context-sensitive formalisms such as Joshi's tree adjoint grammars, which seem sufficient for many natural language constructions, lead to parsers which are still bounded polynomially (although with exponent 6 this time) in the worst case.

An altogether different formalism was proposed more than 50 years ago by Lucien Tesnière as *structural syntax* [48]. This approach developed into the *dependency grammars* of the Russian school of linguistics, and is now used widely

by various computational linguistics researchers. The idea is to axiomatize directly the domination ordering between words, without the mediation of an algebra of syntactic operators. In other words, the trees are not algebraic expressions with words as atomic leaves, the sentence phonetic realisation being the frontier of its constituency structure. They are now graphs whose vertices are the words, arcs denoting their mutual dependencies. Syntactic dislocations will show as crossing dependencies; the dependency graphs may not be represented as planar trees without violating the precedence ordering of the words in the sentence. Dependency graphs are fundamentally different from phrase-structure constituency structures. Note that their size is exactly bounded by the length of the sentence, since every constituent is explicitly anchored on an actual word phonetically realized in the sentence. There is no non-overt copula construct, for instance, a nominal sentence must have a head predicate and a dependent subject. This has advantages – every construction step must consume a terminal – and drawbacks – it is not possible to delay movement in the right argument position by some intermediate computation on the algebraic structure of syntax. Also there is no room for fitting covert quantification.

We remark that the linguistic notion of *head* is extremely overloaded. In dependency grammars, the head of a constituent is the word which has minimal dependency (it labels the root of the tree), whereas in phrase structure formalisms or in minimalism the notion of head is more complex, and is encapsulated in an algorithm that computes inside the constituent in order to project to its head. Thus the head is not a simple uniform notion of the formal apparatus, but rather a specific notion associated with a particular linguistic theory such as $\overline{X}$ theory or minimalism. At the other extreme, a very general framework such as HPSG (Head Phrase Structure Grammars) is rather transparent to linguistic theories, since it consists essentially in first order terms with attribute annotations. It can be considered as a very loose algebraic markup of linguistic data, with all the computing power relegated to the interpreter of the formalism. This neutrality with respect to linguistic theories is good for expressivity, since any Turing complete computational model may be associated with it – the drawback being that it is not logically controled, and thus no complexity analysis is readily available.

2.2 Short Survey of Linguistic Theories of Sanskrit Syntax

In contradistinction to the immense amount of more or less interesting material published in formal general linguistics over the last 50 years, there is a relative scarcity of material available on Sanskrit syntax. Thus Pāṇini's grammar is very extensive on phonology and morphology, but says little on syntax. It however develops a theory of *kārakās* which may be viewed as an interface between syntax and semantics, and is therefore relevant to functional structure. But this theory is sketched only for simple sentences with at most one finite verb, and it is not even clear how it accounts say for relative clauses.[15] Later grammarians

[15] Cardona, Pāṇini: His Work and its Traditions, pp. 167-178 discusses the derivation of complex sentences, and *Recent Research*, 189-197 surveys research on Pāṇinian syntax other than on kāraka relations (P. Scharf).

of the ancient Indian tradition, starting with Bhartṛhari, elaborated further on semantic and pragmatic linguistic analysis, as did Navya Nyāya which introduced a formal language of semiotics in Sanskrit. Other schools such as exegesis (*mīmāṃsā*) developed alternative methods of semantic analysis, but syntax *per se* does not seem to have been a major concern, although systematic ways of paraphrasing linguistic constructions such as compounding may be thought of as transformational theories to a certain extent.

A peculiarity of Sanskrit is the important tradition of commentarial work. Commentaries on terse formulaic styles of teaching (*sūtra*) were necessary for their understanding. Such commentaries deal with several levels of analysis, starting from sandhi viccheda, explained as a prose markup of the original work [50]. The commentarial tradition gives more or less standardized analyses of the commented text, which could be exploited for mechanical extraction. Furthermore, stylistics and poetics was theorized very early by poets such as Daṇḍin and an important Kashmirian school of literary criticism (Ānandavardhana, Abhinavagupta, Mammaṭa, etc.) It is to be expected that this tradition of esthetics, in as much as it influenced the literary production, will be of help ultimately when our computational linguistics tools will be advanced enough to attack the problems of metaphoric expression.[16]

The first general studies of Sanskrit syntax came thus rather late, in India with Apte's manuel [1], and in the West with Speijer treatise [45]. Both build on analogies between Sanskrit and the Western classical languages (Latin and Greek) in order to adapt the then current linguistic terminology to Sanskrit, not entirely convincingly. In these, and most Western Sanskrit grammars until recently, the topic of the proper order of the words in a Sanskrit sentence was dealt with in a rather vague manner; Sanskrit was deemed more or less free-order, but a more detailed analysis was lacking. Besides these two isolated attempts, linguistic studies of Sanskrit consisted mostly of very specialised monographs on specific constructions, often in the context of Vedic literature [39, 49, 19, 20, 21].

The first attempt to give a formal model of Sanskrit syntax was due to Frits Staal in the 60's [46]. What he proposes is a Calder-mobile-like theory of syntax, where oriented trees of first-order logic are replaced by non-oriented trees. In other words, trees which are arguments of a given functor are organised as a multiset rather than as a list. The built-in associativity and commutativity of such a collection allow for arbitrary rotation of arguments, similar to the rotations in a Calder mobile. This gives a precise handle on allowed freedom in ordering of the words of a sentence: any scrambling is allowed, in so far as it is consistent with the domination ordering, in the sense that if a word precedes another, then all the words it dominates must precede the words dominated by the other one. In terms of dependency analysis, there should be no crossing of dependencies. As Gillon expresses it [8]: "[Staal thinks that] word order is free up to preserving constituency."

[16] For references, see Timothy Cahill, An annotated bibliography of the Alamkarasastra, Leiden: Brill, 2001 (P. Scharf).

This immediately gives the limit of the formalism: dislocated sentences cannot be accounted for, unless some prior movement unscrambles long-distance dependencies. Also, it ignores the fact that quantification may be expressed by word order: compare *bālakaḥ gṛhe asti* "the boy is in the house" and *gṛhe bālakaḥ asti* "a boy is in the house".[17] Nevertheless, it gives a simple framework, which may be refined further with dislocation annotations. The crux of the matter is to replace the notion of oriented tree (the trees of concrete syntax and even abstract syntax seen as a construction from constituents ordered by enunciation) by a notion of tree in the algebraic sense of graph theory, where arcs are not necessarily ordered consistently with the enunciation ordering; in other words, the graph may not be representable as a planar graph, with no crossings.

We profit of this view of commutative-associative formalisation of family descriptors to revisit the categorial grammars school. Categorial grammars originated from the work of Ajdukiewicz in the 30's and Bar-Hillel in the 50's, where syntax operators were viewed in analogy with arithmetical fraction notation. The ensueing systems of combinators, with complex notions of raising allowing a more flexible notion of order of evaluation of constituency structure, was simplified by Lambek in the 60's with his L system [41], which allows deduction under hypothesis, and is to categorial combinators what Church's λ-calculus is to Curry combinators [2, 18], or in logic what Gentzen's natural deduction is to Hilbert axiomatic systems [31]. But all the structural rules of sequent calculus are not allowed, and in particular thinning and contraction are not allowed. This is in line with the economy of linguistics, which is different from the economic regime of logic, where hypotheses may be ignored or duplicated, whereas words in a sentence have to be used exactly once in the derivation of its syntactic tree. Furthermore no permutation is allowed either, since word order is assumed to be rigid. Thus Lambek's L system is very close to Girard's linear logic [14], except that Girard's logic is commutative. This connection was noticed in the 90's, when linear logic was firmly established [15]. But if we believe in Staal's model, it is natural to restore commutativity, and try to adapt directly the proof-nets of linear logic to the syntactic descriptions of Sanskrit. We leave this as a hint for future research at this stage.

Let us note further that recent elaborations of dependency grammars add information on the arcs in order to denote functional relationships. For instance, the node headed by some finite form of a transitive verb will typically have as immediate successors two nodes headed by substantival forms, one arc being labeled 'Subject' and one labeled 'Object', or one labeled 'Agent' and the other one labeled 'Goal' if one prefers a more semantic representation. The verbal node may be itself colored as verbal phrase or action. In any case, the structure `[Verb "v" [Subject "s"] [Object "o"]]` may now be twisted as a Calder mobile in order to project its frontier to the enunciation order, be it SOV, SVO, or whatever. Such structures are being used in the Mel'čuk linguistics school in Montreal, in order to represent lexical semantics as semantic graph schemas [35].

[17] This example is borrowed from [16].

2.3 Gillon's Contributions to Sanskrit Syntax

We shall now turn to a more recent attempt at designing a formal notation for Sanskrit syntax, taking into account dislocations. This paragraph will sketch a family of proposals by Brendan Gillon for formalising Sanskrit syntax, consistently with Western theories of linguistics. He made a comprehensive study of Classical Sanskrit syntax, and presented his analyses in a systematic system of notation. This notational system was put to the test on hundreds of characteristic examples from Apte's manual, plus a fair selection from typical sentences extracted from the classical Sanskrit corpus. Let us consider an example. The foundational article is [8], in which he makes a systematic classification of extrapositions in Classical Sanskrit, and in which a first version of his notational system is presented by way of examples. Let us look at a typical one, originally from Kālidāsa's *Mālavikāgnimitra*, given as exemple 2 in Apte's lesson XIV on past participles:

[$_{S}$ [$_{NP6}$ [$_{AP6}$ tayoḥ] baddhayoḥ] [$_{VP}$ 0 [$_{AP1}$ (kim - nimittaḥ)]]
the prisoner what - basis

[$_{NP1s}$ [$_{AP1}$ ayam] __ mokṣaḥ]]
the release

What basis does the release of the two prisoners have?

Gillon explains that the genitive NP6 is extraposed from the subject NP1s. Which he renders by the dislocation notation __. We shall not concentrate on his linguistic analysis, but rather look at the formal descriptive apparatus. The first striking feature is that, despite its phrase-structure early-Chomskian look, it is pretty close to a dependency structure. The NP6 node is indeed an NP6 because it has an immediate constituent which is terminal (it corresponds to the phonetic realisation *baddhayoḥ*), and recognized as an inflected form in the genitive case of a nominal lexical item, namely *baddha*. Thus there is an implicit assumption, namely that each syntax node (a matching pair of square brackets) has an atomic head from which it inherits its features. This assumption is exactly what dependency structures demand: each piece of the construction consumes overt phonetic material. The head, here *baddhayoḥ*, is actually an inflected form whose full morphological tagging is `<N case=6 number=2 gender=? stem="baddha">`. Note how the N and the 6 percolated to its governing left bracket. Note also that we might as well percolate the dual number, which is needed for the gloss 'two prisoners'. Concerning the gender, it is not just undecided, but not needed for the immediate analysis, so I kept it as an unknown. Note also that we made a choice between genitive and locative, a more automated analysis would have to consider the two cases. Finally, a more refined morphological analysis would analyse further *baddha* as a kṛt formation, i.e. past passive participle of root *bandh*. Anyway, such details of granularity notwithstanding, we arrive at a formal structure of the kind [$_{N}$ case=6 number=2 [...] "baddha"], where the substructure [...] stands itself for a similar [$_{N}$ case=6 number=2 [...] "tayoḥ"]. Actually, Gillon labels this substructure *AP*6, meaning that pronoun *tad* is used adjectivally to form the noun phrase 'the two prisoners'. We deem this choice dependent on a

linguistic model where *adjective* is a clear-cut part of speech category. For Sanskrit, the case is not so clear it seems, specially in view of bahuvrīhi compound formation. Actually, *baddha* itself, in the sense of prisoner meant in this sentence, is but the coercion to a substantive of the adjective ‘bound’ issuing from its participial nature. A slightly more primitive analysis would factor together the two nominal forms as $[_N$ case=6 number=2 “tad” “baddha”$]$, leaving to a deeper analysis phase the actual dependency of *tad* on *baddha*.

Let us return to the full example. We note that the subject is indicated by functor $NP1s$. We believe that this should be taken similarly as a feature structure $[_N$ case=1 role=subject ...$]$. But now we have a feature which is not morphological and synthesized from its head, but rather syntactical and inherited from the main clause. This with a particular intended linguistic model recognizing a role for subjects. Actually, another model would recognize this nominative as merely a naming device for the agent, properly carried by the predicate, here the adjective *kiṃnimitaḥ*. This would have the advantage of getting rid altogether of the copula construction $[_{VP}\ 0]$, an anathema for dependency structure since it rests on thin air (0 indicating the lack of phonetic realisation). This construction would be replaced by the predicate marking of *kiṃnimitaḥ*, corresponding to selection of the agent (*kartṛ*). While not strictly speaking Pāṇinian, since Pāṇini restricts kāraka analysis to sentences with finite verbs, this analysis is consistent with Kiparsky’s presentation of the Pāṇinian model [29].

The reader may be confused at this stage, because while I am explaining Gillon’s notation I am proposing to possibly use it in other ways. More precisely, I am suggesting that his meta-formalism could be slightly modified in ways which are suggested by dependency structures, and thus better adapted to automated synthesis, while making it less dependent on a particular linguistic model, with the possible benefit that it could hopefully be used with Pāṇinian models such as Kiparsky’s. What we wish to keep from the meta-notation is the fact that it uses bracketing to denote substructure with percolation of some morphological annotations of their heads, and that a system of references makes explicit the extrapositions/dislocations. In the above example, this suggests replacing the __ place-holder by some integer or label index, with an explicit annotation for its antecedent (here the genitive phrase). This indexing would be similar to λ notation, except that it is linear and bi-directional (i.e. the binder may precede or follow the unique bound occurrence). In other words, this is the information which completes the tree into a dependency graph, crossings being replaced by indirect adressing. What remains a fundamental contribution of [8] is the classification of extrapositions in classical Sanskrit, and the corresponding amendment to Staal’s principle.

Over the years, Gillon modified slightly his notation, and extended it to compound analysis [6, 9, 7, 10, 12, 11, 13].

2.4 Kiparsky’s Analysis of the Pāṇinian Syntax-Semantics Interface

Staal’s monograph dealt mostly with the problem of word ordering in the sentence, but did not account for a number of Sanskrit linguistic constructions,

and stayed at a shallow level of syntactic analysis. His approach was revisited in a joint paper with Kiparsky [30], reprinted in [47]. In this paper, the authors show that Pāṇini's kāraka theory gave a proper account of the interface between shallow syntax (morphological cases) and deep syntax (thematic roles). This investigation was the start of a deeper analysis of Pāṇini's devices and their precise relation with linguistic phenomena such as agreement, ellipsis, anaphora, etc. by Kiparsky, culminating in his survey on the architecture of Pāṇini's grammar [29].

There is not proper space in the present paper to present Kiparsky's analysis in any detail, which anyway appears as a chapter in this volume. But the striking feature, for which Pāṇini departs radically from the usual syntactic treatment of languages such as Latin, French and English, is the elimination of the notion of subject in favor of the notion of agent (more precisely *kartṛ*), and furthermore the attribution of this role, in the case of an active sentence, *not* to the substantival 'subject' (when there is one), but rather to the inflection suffix of the finite verb. Pāṇini's requirement, that every role should have a unique expression in the sentence, deprives the nominative of its elevated status of subject, and makes it a kind of dummy in the underlying semantical binding theory, relegating it to an optional naming role, rather than having it act a proper semantic role. This is quite revolutionary – even if it is anachronistic to say that an ancient grammarian makes a revolutionary proposal to contemporary linguistics! What is most disturbing is that this view shatters the dichotomy between substantivial forms – thought of as having a positive polarity of actors – and verbal forms – thought of as having a negative polarity of roles (i.e. their valency, obtained from their sub-categorization frame by mediation with the voice). Thus the underlying graph matching of positive actors with negative roles, in order to express the semantic constraints, is now replaced by a more symmetric situation calculus, where a sentence may express one of three abstract modes, namely dynamic Action and Reaction, and static Situation. The choice of the mode is conditioned by the verb (in case of verbal sentences), which denotes respectively the Agent, the Goal, or the Process. This model has many advantages. First of all, it gives a proper status to passive forms of intransitive verbs, where an impersonal sentence such as *mayā supyate* lacks a subject, and is not readily translatable in English in a different way than *ahaṃ svapimi*, "I sleep". Secondly, it permits better accommodation of elliptic constructions. If the subject is missing, it is either because it is the deictic you or I when the agentive *tiṅ* denotes the second or first person, and thus optional in a pro-drop language such as Sanskrit, or because it is ellipsed, being the discourse theme, topic or whatever, and is only retrievable from the discourse context.

Kiparsky shows in his survey that this linguistic model accounts for all the linguistic facts such as agreement, provided a proper treatment of co-indexation (corresponding to the Pāṇinian notion of *samānādhikaraṇa*) is effected. This treatment accounts for agreement of verb and subject, as well as concord of substantive and adjective.

We believe that this model is the correct one to analyse the Sanskrit corpus at this functional level which acts as an interface between syntax and semantics.

We further believe that the linguistic constraints can be expressed in an explicit computational model which refines the morphological analysis, and is thus consistent with our requirement of proper embedding of the linear and the functional levels. This computational level must make explicit some hierarchical dependency structure, some complementary dislocation notation, and a proper co-indexation equivalence, needed for agreement/concord as well as corelatives and anaphora. We remark that such co-indexicals appear in more recent variants of Gillon's notation [9].

What is very important is that this functional structure be applied at the level of discourse, rather than at the level of individual sentences. This contextual calculus will be appropriate for instance to analyse a *śloka* in a modular manner, the second half-*śloka* being analysed in the context of the first one, a much needed reduction in complexity. Rather than focusing on the notion of sentence, which will demand global treatment of complex sentences, while not being sufficient for the contextual treatment of anaphora and ellipses, it is much wiser to accommodate directly discourse, and to build a more modular engine able to consider long statements piecewise. Furthermore this "continuous" model accounts easily for chains of absolutives, which typically share their common agent with the main clause. This problem of control is specifically addressed by Gillon in his study of implicit roles in auxiliary clauses [9].

Viewing a text as a continuous discourse, in the tradition of discourse representation structures, has several advantages. Rather than a flat sequence of closed sentence structures $[_S ...]$, we get a sequence of dependent open structures [Agent = x; Goal = y [...] S] where x and y are co-indexicals, the inner [...] is the dependency structure of the current frame, and S is the next frame, in which co-indexicals x and y may appear – x and y are bound in S. Thus [...] may contain co-indexicals x and y, as well as previous ones coming from the context. The scoping of these existentials will be convenient to express quotations: the tool word *iti* marks the closing of some frame, hiding from the rest of the discourse the bindings of the quotation. This structure should also be adequate for representing stories embedded inside stories, in Pañcatantra style. The beginning of a new sub-story will hide the current protagonists, one of which tells the story. When the inner story is finished, its current characters are erased (their scope is ended), and the outer protagonists will pop up again in the discourse context.

In the rest of this section, we shall examine a few recent efforts at computational modeling of the functional level of Sanskrit. We start with our own effort at building a Sanskrit Engine.

2.5 Parsing Sanskrit Text with a Constraint Processor

The method we have been following for parsing Sanskrit text is the following. First of all, we designed a Sanskrit lexicon format accommodating grammatical annotations [24]. Then we implemented internal sandhi and a paradigm-based morphology generator. This morphology generator is applied iteratively to all lexicon entries, yielding a digitalised inflected forms database (implemented as a collection of finite-state automata, one for each lexical category) [22,23]. We then

construct mechanically from this data a modular finite-state transducer inverting external sandhi, using the algorithms developed in [25, 28]. This transducer is used as a non-deterministic tagging segmenter, which we apply to Sanskrit text, preprocessed with the algorithm given below in Appendix B. This lexicon-directed methodology has been sketchily explained in [26].

The next step consists in filtering, out of the possibly thousands of possible segmentations, the few that pass some semantic test inspired from kāraka satisfaction. Basically, this proceeds as follows. For a simple clause with a finite verb form, the valency of the verb[18], together with its voice, determine the cases of nominal phrases which are necessary in order to fulfill the semantic roles demanded by its valency. The sentence is explored in a right to left fashion, looking for the needed phrases. Nominatives are searched with given number and person, thus constraining subjects to agree with their purported verb. Extra nominatives are factored with chosen ones as much as possible, thus building concording nominal phrases (possibly dislocated). Extra oblique cases, as well as extra accusatives, are just ignored as adverbs. Thus instrumentals are deemed needed in passive constructions, as denoting agents, while being ignored in active constructions, as denoting instruments or adverbs of manner. This extremely naive constraint processing is nevertheless quite effective, in filtering out more than 95% of erroneous segmentations in average on a simple training corpus [42], while at least generally retaining the right parse.

An extra mechanism of tag stream transducers, associated with tool words, has been proposed to solve problems such as coordination. Thus 'ca' (in the case it coordinates nominals) operates on the stream of tags of preceding noun phrases, and attempts to replace them with a coordinated tag, effecting proper addition of their numbers (in a non-standard arithmetic model with 3 numbers, 1, 2 and many), as well as proper dominance of their person and gender. This mechanism is sketched in [27], and may be tested as a Sanskrit Reader Companion web service at `http://sanskrit.inria.fr/DICO/reader.html`. We shall in the following refer to this shallow parser as the Sanskrit Engine.

We do not claim that this crude constraint processor is a full-fledged Sanskrit parser. It is merely a nonsense filter, usable as a front-end to a more sophisticated analyser. It does not deal at present with co-relatives, does not attempt to solve anaphora, tends to overgenerate with alleged vocatives, and deals poorly with ellipsed elements. We believe that, specially concerning this last problem, a proper solution involves an analysis closer to Kiparsky's on a continuous discourse flow (as opposed to individual sentences), as sketched at the end of the previous section. Such a more robust parser ought to generate dependency structures annotated with co-indexed semantic roles, in the manner suggested above. Conversely, it ought to be usable as a deterministic checker for a corpus annotated with such representations. It could then learn from a properly annotated corpus (tree bank). It is reasonable to hope that the corpus of Apte's examples

[18] Actually, the valency may depend on the particular verbal form. At present, we use the *gaṇa* of present system forms as a parameter. Thus the verb *pac*, to cook, is marked as transitive for class 1 (*pacati*), and intransitive for class 4 (*pacyate*).

annotated by Gillon, and comprising several hundreds of sentences characteristic of Sanskrit constructions, could be adapted as a training treebank - possibly complemented by longer pieces of text, usable as a discourse treebank. Learning would allow the system to tune its parameters in order to improve its precision, and ultimately yield a deterministic parser usable for fast automatic tagging of digital libraries of a real corpus. Due to the great diversity of styles of classical Sanskrit, specifically tuned parsers, adapted to specific prosodic styles, will probably need to be tuned separately, and meter indications should help the segmenting process.

2.6 Other Efforts at Sanskrit Parsing

Early work on mechanical Sanskrit processing was reported in [51, 38]. We now turn to on-going efforts.

The work reported in [37] on analysing Sanskrit text by the IIT Kanpur team is very promising. Its methodology is actually rather close to the Sanskrit Engine one reported in the previous section. It uses finite-state methods as well. Parsing is guided by a pattern-matching basis of rules with certain priorities. Some guessing is done, instead of exploring systematically all branches of ambiguity. In a certain sense this priority selection algorithm could be seen as a strategy for solving the kāraka constraints. It is not entirely clear how the strategy described agrees with Kiparsky's analysis, though. Impressive exemples of parsed Sanskrit text are given, showing that the approach is sound, even if it cannot yet be used to process satisfactorily a large corpus.

The work reported in [17] presents a Sanskrit tagger, using statistical techniques. It is a POS tagger if you agree on a notion of part of speech which reflects the morphological overt structure, pretty much as assumed in the first part of this paper. Such a statistically trained tagger is an obvious boon for non-deterministic approaches such as the Sanskrit engine methodology. A natural solution would be to incorporate the underlying hidden Markov model in the complete non-deterministic finite-state tagger, in the manner of stochastic automata, in order to guide the search towards the most plausible solution first. Statistical methods are extremely important to tune the performance of linguistic processors, at any level. However, they are not a panacea. They can only cluster the data in categories that are assumed *a priori* by the formal model - one will find lines or shapes in a digital bitmap image only if one looks for them, i.e. measures the data against segment or shape representations. Similarly, a statistical linguistic tool should be seen as an optimizer of formal recognition. Thus statistical methods and algebraico-logical methods have to work hand in hand.

Of course all these approaches need a reliable morphological database. There are currently several teams which have developed significant such databases, for instance in the Sanskrit Library project of P. Scharf and M. Hyman, in the University of Hyderabad project of A. Kulkarni, in the JNU project of G. Jha. The generators are fairly compliant with the Pāṇinian methodology, and thus seem to agree on their common root coverage. This has been checked at least on the noun morphology tables of Kulkarni, Scharf and Hyman, and Huet. A more

systematic evaluation will be possible, when a consensus for a tag set emerges. Further progress towards interoperability will need consensus on root naming standardization. Such naming will specify homonymy indexing, and details such as what should be the main stem of pronouns and numbers.

Other methods have been experimented with, not relying on a full lexicon. This has the advantage of not requiring at the start a full coverage of the target corpus' vocabulary by the morphological closure of the root lexicon. But the problem of lemmatizing a conjectured form without a handle on possible stems is very hard. It involves undoing internal sandhi, a non-rational transducing problem in view of retroflexion, thus not easily amenable to finite state methods. Furthermore, if one wants to do segmentation at the same time, one will have to face undoing external sandhi modulo undoing internal sandhi, a problem very hard to solve exactly, although heuristics may help in a fair number of cases.

2.7 Designing a Functional Level Formalism

This paper has sketched a general framework for representing Sanskrit text. The main framework uses discourse representation structures, in order to manage scoping and naming of indexicals. The functional structure may be thought of as a situation calculus, in the tradition of Montague semantics (higher order type theory in the sense of Church, possibly enriched as a dependent type framework in the sense of Martin-Löf [34]). Indexicals will serve to group together actors of the same function in a phrase, sharing the features attributed to the forms it groups. This gives the treatment of both agreement and concord. Note that sharing may affect a gender feature to a verbal form, if it has an overt subject in the active voice (respectivally an overt object in the passive voice).

It should be understood that the functional level, where kāraka satisfaction takes place, is *not* yet the semantic level. For instance, it does not need to specify the analysis of compounds. The only really necessary distinction among compounds is between *avyayībhāva* (adverbials) and *tatpuruṣa/karmadhāraya*, which play a unique role, consistent with their case endings. There is no real need to explicitate the possible adjectival role of a bahuvrīhi compound. Rather, such a compound will be co-indexed with other concording substantives. It is left to a further level of semantics to figure out the needed coercions between individuals, characteristic properties and classes which are needed to make sense of the gliding between adjectival and substantival uses.

Similarly, genitives do not really need to be explicitly linked to the actual word they complement. They can "float around" so to speak, without clear dependency or constituency specification. This is consistent with Pāṇini, who does not associate a kāraka with genitives. Deciding the role of genitives in a phrase may thus be left to a semantic layer, where common sense reasoning (i.e. hard deduction) might be needed for their disambiguation. For instance, even a very simple sentence [42] such as: *tiṣṭhan bālaka upādhyāyasya praśnānām uttarāṇi kathayati* seems to require for its understanding that answers belong to questions, and that questions belong to the teacher.

In the same vein, rather than requiring a dative argument in the subcategorisation frame of verbs such as *dā*, it may be better to consider it a complement rather than a needed argument. Thus the functional modeling may operate at various granularity levels, between a crude consistency check and a deeper semantic analysis, and consequently subcategorization may be more or less coarse. Nevertheless, we may expect problems at the right handling of verbs with two accusatives, since knowing which is which often demands some common sense reasoning.

What distinguishes clearly the functional level from the linear level is that it is branching in a tree-like constituency, or rather dependency structure. Thus in addition to the total ordering of enunciations (i.e. the temporal sequence of the phonemic chunks), there is a partial ordering of dependencies, where bracketing indicates scoping. Thus for instance absolutive clauses depend on the rest of the sentence, with at the top the main finite verbal phrase, which itself depends on the surrounding discourse. Here dependency runs contrary to enunciation, so that the intermediate situations may get their ellipsed arguments from the future to a certain extent. We see this dependency structure as the analysis of discourse, linking situations in some kind of discourse Montagovian semantics. A particle such as *iti* is typically a discourse operator, closing a quotation, i.e. a sub-discourse, which may span several sentences. Similarly, the *daṇḍa* punctuation symbol should be treated as a discourse operator, allowing the left half-*śloka* to be analysed piecewise, while having access to roles expressed in the second half. In contrast, the double danda is a fuller break in the discourse. Having an explicit handle on the discourse will facilitate the analysis of deictics through an explicit representation of elocution situations ('I' refers to the speaker in the current sub-discourse, and its gender/number consistency may be checked, etc.)

We see kāraka satisfaction following Kiparsky's analysis, with the selection of a head, its coercion to one of the three modes, and the verification that roles are fulfilled, by co-indexed complexes visible in the discourse dependency structure. The proof of consistency may or may not be explicit in the functional structure, there may be several levels of partial specification. In case of participial heads, the lexicon must give an indication of their prefered voice – Passive for passive participial forms, for instance. The search for a consistency proof, when the corpus is tagged only with the dependency bracketing, will look for roles in the appropriate periphery of their predicate, according to analyses such as Gillon's on Bhartṛhari's rule [9].

We stop here our speculations about the precise specification of the functional level, since as we emphasized there is a space for design which needs concrete experimentation to validate precise data structures and algorithms. As already said, we are still far from a semantic level, where the notation will express logical deductions which are not verbalized, and thus by necessity the algebraic structure is richer than just a dependency graph between phonetic realisations. Formalisms are ready to be used for this modeling, through the Montague semantics, which has been revived lately with material coming from the denotational semantics of programming languages. We have rich type theories in which to embed logical

deductions and reasoning in ontological structures with grammars of descriptions, etc, etc. Nonetheless, all this logical tradition is still a little bit naive with respect to understanding natural language, it somehow assumes some universal structure of thoughts and concepts on which natural language productions should project, once the syntax is deciphered. But there are so many intermediate levels which can be considered. For instance, style. Style is a fundamental parameter which should accompany the discourse structure. Here actually, we are rather fortunate in Sanskrit, since there is a very important tradition of literary criticism and esthetics of poetry. This tradition spells out particular styles, such as more or less intensive use of compounds. Corpus material which obeys the canons of style praised by the masters will tend to conform to certain recognizable patterns, thus helping the understanding of the text by mechanical means. Furthermore a lot of *śāstra* literature is itself strongly structured; for instance, the *Dhvanyāloka* consists in a dense teaching in a form of terse *kārikā* verses, interwoven with a prose commentary (*vṛtti*), which is itself commented upon by further authors. Proper understanding of such texts will need a precise grasp on this commentary structure [50]. We get discourse within discourse in similar ways as the *Pañcatantra* tells us tales within tales, and this suggests a continuum between the discourse structure within one text and the intertextual structure of a Sanskrit digital library.

A closely related phenomenon is prosody. Knowing the meter of metrical poetry will be a tremendous help to the segmenter. Knowing the accent will permit to recognize bahuvrīhi compounds. The enormous Indian tradition of prosody, *chandas*, has to be exploited for proper computational modeling. Finally, coming to semantics, it is not so clear that the Western logical apparatus which is the substrate of theoretical investigations of semantics in Western linguistics is entirely appropriate for Sanskrit. It corresponds more or less to a mathematical abstraction of *prācīnanyāya*, 'old' logic. Whereas Indian investigations in the theory of language, by Gaṅgeśa and his successors, developed a new logic *navyanyāya* which is a kind of sophisticated formal semiotics. Furthermore, there is an important Indian tradition of pragmatics, arising from the philosophical school of *mīmāṃsā*. When speech is ritualised, it means it contains overt structure that may be recognized. Mantras are not random nonsense, they are actually strongly structured patterns [52]. If we take the extreme view that speech is driven by communication intention, then meaning must depend on will. We have to look for the fruit of the speech act, in pretty much the same way that the policeman looks for the motive of the criminal.

2.8 A Roadmap for Future Work

The task is formidable indeed, and even if all investigating teams join forces in collaborating on open standards for interoperability of the various available tools, it will take many years to build Sanskrit processing computational platforms that will process usefully a significant part of a giant corpus, and we must be modest in our goals. But it seems we can at least sketch a roadmap of plausible stages.

One problem that has been only tangentially addressed is the structure of the lexicon. This is however of paramount importance, whatever methodology is followed. The problem of completeness of the lexicon is a very complex one, since the computerized lexicon need not only be a static data base of stems and forms. It is more appropriate to look at the lexicon as a generative device - for instance *kṛt* and/or *taddhita* formation could be expressed implicitly as a grammar, and dictionary lookup implemented as parsing.

Another problem is the exact representation of parameters, both for morphology (e.g. intercalating *i* in the future stem, *gaṇa* class of verbs, *parasmaipada* vs *ātmanepada* use, etc.) and for the functional structure (subcategorization regime of verbs, etc.) A close look at the information encoding in *dhātupāṭhas* and *gaṇapāṭhas* is a good starting point. More parameters will progressively percolate from the constraint satisfaction algorithms. The general point is that all this information ought to be lexicalized, rather than being hard-wired in the processing programs.

Still another problem is how to organize the diachronic character of a language that spans 3 milennia and one subcontinent. The combinatorics of Vedic are likely to be different from the ones of Classical Sanskrit, the vocabulary is differently distributed, the morphological operations have evolved - some tenses have vanished, while compound formation increased, and prepositions got more rigidly affixed as preverbs. This is clear from Pāṇini's grammar, which goes at length to describe exceptions in order to cover the attested Vedic corpus. It seems that the digital lexicon ought to have diachronic annotations which will turn on or off certain generative processes, according to the intended corpus. This area of research has hardly been touched yet it seems (but see [36] for epic Sanskrit, also [43] for Vedic).

The basic assumption of the two-level model which we presented is that the linear treatment (tagging) is complete, so that we may make the more semantic analysis without need to backtracking into the search space for segmentation/lemmatisation. We remind the reader that quality of non-deterministic search to solve any problem is expressed by two parameters: precision and recall, measuring respectively the lack of extraneous solutions, and the covering of the right solutions by the search space. Poor precision means many answers are returned, there is noise. Poor recall is worse, there is silence - i.e. a legitimate enunciation is not recognized. Good design means no silence, and noise minimised. Typically, several answers are potentially produced, but the intended one comes very early in the list. If it always come first, then we may use it as a function, determinism is restored. But this ideal situation is not always possible, since we know there are actual ambiguities, such as *śvetodhāvati*, where the white (horse) coexists so to speak with a dog, in the absence of a disambiguating context.

We claim that segmentation/tagging is complete in a strong sense - every Sanskrit sentence has a finite number of interpretations modulo *sandhiviccheda*

and morphological analysis. Thus, if the lexical basis is sufficient to cover the vocabulary of the target corpus, we are left with an optimisation problem, of computing the most probable functional/semantic structure in the finite space of all possible taggings. This is where statistical methods enter into play. We shall look optimally for the solution in choosing the one maximizing some likeliness criterion in the stochastic functional space.

In order to train the statistical optimizer, we need to build a treebank for learning its parameters. Building a treebank involves two steps. First, a complete specification of the datastructure for the functional structure, since its formal design has been only sketched above. Secondly, one must prepare a fully tagged corpus, an expensive task, since it involves tedious work by a human expert - someone understanding both Sanskrit, linguistics, and informatics. One obvious path to take would be to build on the work of Brendan Gillon's tagging of Apte's sentences, by adapting his parses to a non-ambiguous representation of dependencies and indexicals. However, we shall need an explicit schematic representation of the context, in order to get the right discourse structure, and not just successive parses of sentences. Once a sufficient training basis will be available, the system should be able to bootstrap, since we shall be able to use the functional satisfaction algorithm on untagged corpus, in order to present a small number of potential structures to the human expert analysing raw corpus, and thus considerably speed up the treebank production. Specialized treebanks will be used to train specialised recognizers, for various styles, various periods, etc. Failures of proper recognition will turn into opportunities for lexicon acquisition. Here problems of maintenance and versioning of the lexical database, in a possibly distributed manner, will arise.

At some point in the future, when available software will be robust enough to be usable for philological applications, the need will arise to inter-operate with digital library formats, such as Perseus [`http://www.perseus.tufts.edu/`] or IntraText [`http://www.intratext.com/`].

Conclusion

We have argued in this paper that a formal linguistic model for Sanskrit, usable for computational treatment and ultimately understanding by computers, should distinguish 3 levels, called linear, functional and semantical. The linear level is appropriate to treat by finite-state methods the morpho-phonetic processes of inflexion, sandhi, and their inverses segmentation and lemmatisation. The functional level is seen as dependency graphs applied to discourse structure. This level should make explicit the scoping structure and a co-indexation relation, sufficient to express and or verify control such as kāraka consistency. This functional level has not been fully specified, since it may be more or less detailed, but its design space has been sketched, and it has been argued that parsing prototypes such as [37, 27] operate on this level. The semantical level is

left for long term research, since it involves complex common sense reasoning, the use of metaphors and other figures of style, all things which a computer is notoriously bad at. Thus a roadmap has been sketched for future research, in which many opportunities of cooperation by the various teams working on computational Sanskrit linguistics will arise.

References

1. Apte, V.S.: The Student's Guide to Sanskrit Composition. In: A Treatise on Sanskrit Syntax for Use of Schools and Colleges. Lokasamgraha Press, Poona (1885)
2. Barendregt, H.: The Lambda Calculus: Its Syntax and Semantics. North Holland, Amsterdam (1984)
3. Bharati, A., Chaitanya, V., Sangal, R.: Natural Language Processing. A Paninian Perspective. Prentice-Hall of India, New Delhi (1995)
4. Dowty, D.: Grammatical relations and Montague Grammars. In: Jacobson, P., Pullum, G.K. (eds.) The nature of Syntactic Representation, Reidel (1982)
5. Eilenberg, S.: Automata, Languages, and Machines, volume A. Academic Press, London (1974)
6. Gillon, B.S.: Bartṛhari's solution to the problem of asamartha compounds. Études Asiatiques/Asiatische Studien 47(1), 117–133 (1993)
7. Gillon, B.S.: Autonomy of word formation: evidence from Classical Sanskrit. Indian Linguistics 56(1-4), 15–52 (1995)
8. Gillon, B.S.: Word order in Classical Sanskrit. Indian Linguistics 57(1), 1–35 (1996)
9. Gillon, B.S.: Bartṛhari's rule for unexpressed kārakas: The problem of control in Classical Sanskrit. In: Deshpande, M.M., Hook, P.E. (eds.) Indian linguistic studies: Festschrift in honour of George Cardona, Motilal Banarsidass, Delhi (2002)
10. Gillon, B.S.: Null arguments and constituent structure in Classical Sanskrit. Private communication (2003)
11. Gillon, B.S.: Subject predicate order in Classical Sanskrit. In: Scott, P., Casadio, C., Seely, R. (eds.) Language and grammar: studies in mathematical linguistics and natural language, pp. 211–225. Center for the Study of Language and Information (2005)
12. Gillon, B.S.: Exocentric (bahuvrīhi) compounds in classical Sanskrit. In: Huet, G., Kulkarni, A. (eds.) Proceedings, First International Symposium on Sanskrit Computational Linguistics, pp. 1–12 (2007)
13. Gillon, B.S.: Pāṇini's aṣṭādhyāyī and linguistic theory. J. Indian Philos 35, 445–468 (2007)
14. Girard, J.-Y., Lafont, Y., Régnier, L. (eds.): Advances in Linear Logic. London Mathematical Society Lecture Notes, vol. 222. Cambridge University Press, Cambridge (2005)
15. Girard, J.-Y., Lafont, Y., Taylor, P. (eds.): Proofs and Types. Cambridge Tracts in Theoretical Computer Science, vol. 7. Cambridge University Press, Cambridge (1988)
16. Goyal, P., Sinha, R.M.K.: Translation divergence in English-Sanskrit-Hindi language pairs. In: Kulkarni, A., Huet, G. (eds.) Sanskrit Computational Linguistics. LNCS, vol. 5406, pp. 134–143. Springer, Heidelberg (2009)

17. Hellwig, O.: SanskritTagger, a stochastic lexical and pos tagger for Sanskrit. In: Huet, G., Kulkarni, A. (eds.) Proceedings, First International Symposium on Sanskrit Computational Linguistics, pp. 37–46 (2007)
18. Hindley, J.R., Seldin, J.P. (eds.): Introduction to Combinators and λ-Calculus. Cambridge University Press, Cambridge (1986)
19. Hock, H.H.: The Sanskrit quotative: a historical and comparative study. Studies in the Linguistic Sciences 12(2), 39–85 (1982)
20. Hock, H.H. (ed.): Studies in Sanskrit Syntax. Motilal Banarsidass, Delhi (1991)
21. Hoffmann, K.: Der Injunktiv im Veda. Eine synchronische Untersuchung. Karl Winter Universitätsverlag (1967)
22. Huet, G.: The Zen computational linguistics toolkit: Lexicon structures and morphology computations using a modular functional programming language. In: Tutorial, Language Engineering Conference LEC 2002 (2002)
23. Huet, G.: Towards computational processing of Sanskrit. In: International Conference on Natural Language Processing (ICON) (2003), `http://yquem.inria.fr/~huet/PUBLIC/icon.pdf`
24. Huet, G.: Design of a lexical database for Sanskrit. In: Workshop on Enhancing and Using Electronic Dictionaries, COLING 2004. International Conference on Computational Linguistics (2004), `http://yquem.inria.fr/~huet/PUBLIC/coling.pdf`
25. Huet, G.: A functional toolkit for morphological and phonological processing, application to a Sanskrit tagger. J. Functional Programming 15(4), 573–614 (2005), `http://yquem.inria.fr/~huet/PUBLIC/tagger.pdf`
26. Huet, G.: Lexicon-directed Segmentation and Tagging of Sanskrit. In: Tikkanen, B., Hettrich, H. (eds.) Themes and Tasks in Old and Middle Indo-Aryan Linguistics, pp. 307–325. Motilal Banarsidass, Delhi (2006)
27. Huet, G.: Shallow syntax analysis in Sanskrit guided by semantic nets constraints. In: Proceedings of the 2006 International Workshop on Research Issues in Digital Libraries. ACM, New York (2007), `http://yquem.inria.fr/~huet/PUBLIC/IWRIDL.pdf`
28. Huet, G., Razet, B.: The reactive engine for modular transducers. In: Futatsugi, K., Jouannaud, J.-P., Meseguer, J. (eds.) Algebra, Meaning, and Computation. LNCS, vol. 4060, pp. 355–374. Springer, Heidelberg (2006), `http://yquem.inria.fr/~huet/PUBLIC/engine.pdf`
29. Kiparsky, P.: On the architecture of Pāṇini's grammar. In: International Conference on the Architecture of Grammar, Hyderabad (2002)
30. Kiparsky, P., Staal, J.F.: Syntactic and semantic relations in Pāṇini. Foundations of Language 5, 83–117 (1969)
31. Kleene, S.C.: Introduction to Metamathematics. North Holland, Amsterdam (1971) (8th reprint (1st edn. 1952))
32. Kracht, M.: The combinatorics of case. Research on Language and Computation 1(1/2), 59–97 (2003)
33. Kulkarni, M.: Phonological overgeneration in paninian system. In: Huet, G., Kulkarni, A., Scharf, P. (eds.) Sanskrit CL 2007/2008. LNCS (LNAI), vol. 5402, pp. 306–319. Springer, Heidelberg (2009)
34. Löf, P.M.: Intuitionistic Type Theory. Bibliopolis, Napoli (1984)
35. Mel'cŭk, I.: Dictionnaire explicatif et combinatoire du français contemporain. Recherches lexico-sémantiques IV. Les Presses de l'Université de Montréal (1999)

36. Oberlies, T.: A Grammar of Epic Sanskrit. De Gruyter, Berlin (2003)
37. Pawan Goyal, V.A., Behera, L.: Analysis of Sanskrit text: Parsing and semantic relations. In: Huet, G., Kulkarni, A. (eds.) Proceedings, First International Symposium on Sanskrit Computational Linguistics, pp. 23–36 (2007)
38. Ramanujan, P.: Computer processing of Sanskrit. In: Computer Processing of Asian Languages Conference 2. IIT Kanpur (1992)
39. Renou, L.: La valeur du parfait dans les hymnes védiques. Honoré Champion, Paris (1925); 2ème édition étendue (1967)
40. Renou, L.: Terminologie grammaticale du sanskrit. Honoré Champion, Paris (1942)
41. Rétoré, C.: The logic of categorial grammars. Technical report, INRIA Rapport de recherche 5703 (2005), http://www.inria.fr/rrrt/rr-5703.html
42. Sastri, V.: Samskrita Bālādarśa. Vadhyar, Palghat (2002)
43. Scharf, P.: Pāṇinian accounts of the vedic subjunctive. Indo-Iranian Journal 48(1-2), 71–96 (2005)
44. Scharf, P., Hyman, M.: Linguistic Issues in Encoding Sanskrit. Motilal Banarsidass, Delhi (2009)
45. Speijer, J.S.: Sanskrit Syntax. E. J. Brill, Leyden (1886)
46. Staal, J.F.: Word Order in Sanskrit and Universal Grammar. Reidel, Dordrecht (1967)
47. Staal, J.F.: Universals - Studies in Indian Logic and Linguistics. The University of Chicago Press (1988)
48. Tesnière, L. (ed.): Éléments de Syntaxe Structurale. Klincksieck, Paris (1959)
49. Tikkanen, B.: The Sanskrit Gerund: a Synchronic, Diachronic and typological analysis. Finnish Oriental Society, Helsinki (1987)
50. Tubb, G.A., Boose, E.R.: Scholastic Sanskrit. Columbia University, New York (2007)
51. Verboom, A.: Towards a sanskrit wordparser. Literary and Linguistic Computing 3, 40–44 (1988)
52. Yelle, R.A.: Explaining mantras. Routledge, New York (2003)

Appendix A: Phonemes from Classical Sanskrit

The various transliteration schemes are designated as VH for the adaptation of the Velthuis scheme used in [27], KH for Kyoto-Harvard, WX for the Hyderabad-Tirupati scheme, SL for the one used by the Sanskrit Library effort.

In the table below, Code 14 is anusvāra, indicated by *bindu*, while code 15 is *anunāsika*, indicated by *candrabindu*. Code 50 is used to note the hiatus. It is needed for schemes VH and KH, since they are not prefix codes. Thus *a_i* is needed to represent the hiatus without confusion with *ai*. Similarly for *a_u*, for instance in word *tita_u*. See Appendix B for another use of this code, independently from transliteration.

Vedic *l* is not accommodated at this point. It is proposed to reserve code 0 for hyphen and code -1 for avagraha. Other codes will be needed for extra punctuation, numerals, the marking of accent, optional marking of proper names, vocatives and other interjections, etc.

code	*deva*	*roma*	*VH*	*KH*	*WX*	*SL*	*code*	*deva*	*roma*	*VH*	*KH*	*WX*	*SL*
1	अ	a	`a`	`a`	`a`	`a`	26	ञ्	ñ	`~n`	`J`	`F`	`Y`
2	आ	ā	`aa`	`A`	`A`	`A`	27	ट्	ṭ	`.t`	`T`	`t`	`w`
3	इ	i	`i`	`i`	`i`	`i`	28	ठ्	ṭh	`.th`	`Th`	`T`	`W`
4	ई	ī	`ii`	`I`	`I`	`I`	29	ड्	ḍ	`.d`	`D`	`d`	`q`
5	उ	u	`u`	`u`	`u`	`u`	30	ढ्	ḍh	`.dh`	`Dh`	`D`	`Q`
6	ऊ	ū	`uu`	`U`	`U`	`U`	31	ण्	ṇ	`.n`	`N`	`N`	`R`
7	ऋ	ṛ	`.r`	`R`	`q`	`f`	32	त्	t	`t`	`t`	`w`	`t`
8	ॠ	ṝ	`.rr`	`RR`	`Q`	`F`	33	थ्	th	`th`	`th`	`W`	`T`
9	ऌ	ḷ	`.l`	`L`	`L`	`x`	34	द्	d	`d`	`d`	`x`	`d`
10	ए	e	`e`	`e`	`e`	`e`	35	ध्	dh	`dh`	`dh`	`X`	`D`
11	ऐ	ai	`ai`	`ai`	`E`	`E`	36	न्	n	`n`	`n`	`n`	`n`
12	ओ	o	`o`	`o`	`o`	`o`	37	प्	p	`p`	`p`	`p`	`p`
13	औ	au	`au`	`au`	`O`	`O`	38	फ्	ph	`ph`	`ph`	`P`	`P`
14	ं	ṃ	`.m`	`M`	`M`	`M`	39	ब्	b	`b`	`b`	`b`	`b`
15	ँ	ṁ	`"m`	`M`	`z`	`~`	40	भ्	bh	`bh`	`bh`	`B`	`B`
16	ः	ḥ	`.h`	`H`	`H`	`H`	41	म्	m	`m`	`m`	`m`	`m`
17	क्	k	`k`	`k`	`k`	`k`	42	य्	y	`y`	`y`	`y`	`y`
18	ख्	kh	`kh`	`kh`	`K`	`K`	43	र्	r	`r`	`r`	`r`	`r`
19	ग्	g	`g`	`g`	`g`	`g`	44	ल्	l	`l`	`l`	`l`	`l`
20	घ्	gh	`gh`	`gh`	`G`	`G`	45	व्	v	`v`	`v`	`v`	`v`
21	ङ्	ṅ	`"n`	`G`	`f`	`N`	46	श्	ś	`"s`	`z`	`S`	`S`
22	च्	c	`c`	`c`	`c`	`c`	47	ष्	ṣ	`.s`	`S`	`R`	`z`
23	छ्	ch	`ch`	`ch`	`C`	`C`	48	स्	s	`s`	`s`	`s`	`s`
24	ज्	j	`j`	`j`	`j`	`j`	49	ह्	h	`h`	`h`	`h`	`h`
25	झ्	jh	`jh`	`jh`	`J`	`J`	50	_	_	`_`	`_`	`_`	`_`

Appendix B: Input Preprocessing

We sketch an algorithm for transforming a list of chunks separated by blanks into a list of proper forms fit for segmentation processing. The specification is the following. We want the input list of chunks to be obtainable from the phonemic form of the continuous saṃhitapāṭha of the considered Sanskrit text by mere insertion of blanks at the sandhi junction of the words (or at the necessary initial spacing in case of hiatus). We want the proper forms to be in final sandhi. For instance, the padapāṭha is a list of such proper forms, but it is not necessary that a proper form consists of just one *pada*, and indeed e.g. the form *tacchrutvā* is proper. The form *viṣṇo* is a proper (vocative) form, whereas *devo* is not, although it is in final sandhi form as a string of phonemes, but the proper (nominative) form is *devaḥ*.

Finally, we want the algorithm to be deterministic, while being complete. That is, we want all segmentation solutions of the candidate sentence to be obtainable by concatenating individual solutions to the separate segmentation of the proper forms. Thus all non-deterministic search is done within the sandhi

segmentation of the separate proper forms, but the production of these forms is a deterministic preprocessing. This last requirement poses a problem for certain hiatus situations, since a chunk ending in vowel *a* preceding a chunk starting with say vowel *u* could come by hiatus from a stem ending in *aḥ* as well as from a stem ending in *e*, as well as from a stem ending in *o*. In order to solve this dilemma, we shall not process these two chunks *... a u ...* as two separate forms, but rather keep them glued within one form *... a_u ...*, where the underscore indicates an explicit hiatus situation which will be handled by the sandhi splitter in a non-deterministic fashion. This is the reason why we added code 50 above in our tables, even though it is not a phoneme proper, but a glottal stop. We want to emphasize that this hiatus symbol is necessary in the internal representation of sandhi and its inversion, while it is neither necessary nor useful in the external input form – chunks do not contain underscores – except that they are needed exceptionally in order to input words which have internal hiatus such as *pra_uga* when the transliteration scheme is not a prefix code (this is the case for the VH and KH schemes above, whereas WX and SL are prefix codes, and indeed express a same-length transduction, admittedly a slight advantage). A final remark is that the full padapāṭha presentation is *not* in general a *bona fide* input form, since for instance neither *kutrāpi* nor *anyokti* nor *tacchrutvā* are decomposable into chunks obeying our requirements.

Let us now turn to the algorithm. We shall describe it in phases. It takes as input a list of chunks, and produces as output a list of proper forms. When the input is the empty list, so is the output. When the input is a unit chunk, we return it as a unit list of forms. Now assume the input is the non-unit list `[ chunk :: chunklist ]`, where we have recursively pre-processed `chunklist` into a non-empty list of proper forms `formlist`. Let `c` be the first character of the first form in `formlist`. In the general case, given `c` we adjust `chunk` into a proper form `form`, and we return the list `[ form :: formlist ]`. This adjustment is by cases on the last character `x` of `chunk`. If `x` is anusvāra, `form` is `chunk` where the last character is replaced by *m*. If `x` is *o*, then we replace it by the sequence *aḥ* whenever `c` changes accordingly this sequence into *o* by sandhi. This includes the possible case when it is avagraha, in which case it is replaced by *a* in the output form. If `x` is *d, n, c* or *l*, then we revert it to *t* whenever `c` changes accordingly *t* into `x` by sandhi. Finally, when `x` is *a*, if `c` is not a vowel different from *a* there is no adjustment, otherwise we return `formlist`, where the first form is changed by prefixing it with `chunk` followed by underscore. Similarly, when `x` is *ā*, if `c` is not a vowel there is no adjustment, otherwise we return `formlist`, where the first form is changed by prefixing it with `chunk` followed by underscore, recognizing again a hiatus situation. To complete the algorithm, we only need to take care to rewrite an initial avagraha into its original *a*. The avagraha notation is necessary, by the way, as we shall see.

The reader will have understood that our algorithm implements simple cases of sandhi inversion, in order to interpret for instance the chunks list *tad api* as the forms list *tat api*, *tan matra* as *tat matra*, *tac ca* as *tat ca*. Thus the sentence *yad iha asti tad anyatra yan neha asti na tat kvacit* may be understood as a

chunks list, and preprocessed to the forms list *yat iha asti tat anyatra yat neha asti na tat kvacit* where now each of the 11 forms may be separately un-sandhied, for instance using the algorithm given in [25], in order e.g. to segment further *neha* into the primitive forms *na iha*. Each of the forms is in terminal sandhi, like *tat*, which may be recognized as the final sandhi form of either the nominative or the accusative singular of the neuter pronoun *tad*. However, remark that we do the transformations only in cases where it is safe to be applied in a deterministic fashion. For instance, the *subhāṣita* input as the list of 8 chunks: *na hi kukkuṭyā ekadeśaḥ pacyata ekadeṣaḥ prasavāya kalpate* is preprocessed into the list of 7 forms *na hi kukkuṭyā ekadeśaḥ pacyata_ekadeṣaḥ prasavāya kalpate* so that it is left to the non-deterministic unsandhier to select among the 2 potential solutions of sandhi segmentation of the composite form *pacyata_ekadeṣaḥ* the right one, viz. *pacyate ekadeṣaḥ*. Note that underscore acts nonetheless as a segmentation hint, no segmentation overlapping with underscore will be attempted.

Maybe some extra argumentation of the prediction of *o* as *aḥ* in favorable contexts is needed. Care must be taken for completeness, since after all *o* is a permitted final, of vocative forms of *u* stems for instance. Thus the (admittedly ad-hoc) saṃhitapāṭha sentence *viṣṇodadhidadhyāt* may only be presented in padapāṭha form as the two chunks *viṣṇodadhi dadhyāt* but definitely not as the three chunks *viṣṇo dadhi dadhyāt*, since the preprocessing would yield the fautive form **viṣṇaḥ*. This is implied from the fact that sandhi of *aḥ* and *d* yields *od*. Furthermore, if one wants to force a segmentation hint at the juncture of *viṣṇo* and *dadhi*, it is easy to allow underscore to appear in the form *viṣṇo_dadhi*. All is needed is to add to the finite-state description of sandhi a few extra rules such as *o+d* →*o_d*, and the sandhi splitting will do the segmentation correctly into the forms *viṣṇo* and *dadhi*. Note that this does not overgenerate, since no other sandhi rules may produce *o_d*. Finally, an adaptation of this example to an initial *a* yields *viṣṇo'mṛtaṃdadhyāt*, which may be presented as the chunks list *viṣṇo'mṛtaṃ dadhyāt*. This example shows the necessity of the avagraha convention for completeness. Finally, we remark that anusvāra coming from sandhi of final *n* is *not* assimilated to ṃ by preprocessing, since it is followed by a sibilant and thus it is not at a place legal for segmentation. Consider for instance *devāṃśca*, which we cannot present as *devāṃ śca* any more than we can break *tacchrutvā*.

The preprocessing algorithm is very useful because it allows easy insertion of segmentation hints by placing extra spaces, without having to rewrite say *amṛtaṃdadhyāt* into the two final sandhi forms *amṛtam dadhyāt*: preprocessing will do that for you automatically from the chunks list *amṛtaṃ dadhyāt*.

Although this proprocessing seems to be consistent with the usual ways of presenting Sanskrit text in transliteration, there is no uniform convention, especially concerning the use of avagraha. Very often texts for students follow rather a padapāṭha way of presentation, so that beginners don't face the full complexity of sandhi. In such texts [42] we find: *kṛṣṇaḥ uttiṣṭhatu* instead of *kṛṣṇa uttiṣṭhatu*. Indeed such notation eases reading for the beginner, since it prevents his trying all segmentations with forms *kṛṣṇe* and *kṛṣṇo*. This difficulty

turns into overgeneration of our Sanskrit processing tools. Even if the segmenter is driven by a lexicon of inflected forms, so that he will not consider *kṛṣṇo*, he will still have to consider *kṛṣṇe* besides *kṛṣṇaḥ*. These 2 choices, when cumulated in a long sentence, yield exponential behaviour in the number of chunks in hiatus situations. Furthermore, elimination of the *kṛṣṇe* branch will have to do some semantic evaluation, for all the morphological solutions (in this case, the 9 different morphological productions of *kṛṣṇe* from *kṛṣṇa* in the 3 genders). This is indeed a heavy price to pay. Of course, if we accept entering *kṛṣṇaḥ uttiṣṭhatu*, the segmentation will be unique. Note however that the form with vocative *kṛṣṇa* is not presentable with our convention as: *kṛṣṇa uttiṣṭha* (which would correspond actually to wrong padapāṭha as *kṛṣṇe* | *uttiṣṭha*). It must be presented in sandhied form as *kṛṣṇottiṣṭha*. But if we want to allow the absence of sandhi after an initial vocative, we have to invent some extra separator such as | in order to accept as input *kṛṣṇa* | *uttiṣṭha*. This separator will then allow a less ambiguous padapāṭha input, specially if it is completed by a similar convention for compound viccheda.

Similarly, remark that the part of our preprocessor which infers *yat* from chunk *yan* preceeding chunk *neha* is only an incomplete heuristics, which will prevent to put a space after a genuine *n* form such as *brahman* in a similar context. Note also that chunking such as *ity ukta* for *ityukta* is not recognized. In our interface we allow such chunking (as well as similar transformation of *u* into *v* before vowels), but in this case the effect of the preprocessor is just to glue the two chunks into one, losing the benefit of the segmentation information. The tradeoff here is between glueing for completeness, but not profiting of the segmentation hints, and applying some incomplete un-sandhi heuristics, obliging to forbid spaces in the cases where solutions would be missed. There is no completely optimal solution to such tradeoff. The most reasonable attitude at this point is to evaluate variations of the chunking preprocessing, in order to determine what is the best fit on a representative sample of the existing digitalised corpus. The transliteration parameter is irrelevant, since going from one column to another in Table I is an easy rational transduction, which can normalize a corpus in whatever phonemic transliteration or syllabic notation, in whatever encoding, ASCII or Unicode-UTF8 or whatever. The only difficulty will be to give codes to extra notation, dandas, other punctuation symbols, vedic letters, numerals, accents, prosody, etc. If some consensus can be reached for a few digital libraries, then the preprocessing of spaces and *avagraka*-like notations may be quickly standardized, allowing the design of inter-operable software.

Analysis of Sanskrit Text: Parsing and Semantic Relations

Pawan Goyal[1], Vipul Arora[1], and Laxmidhar Behera[1,2]

[1] Indian Institute of Technology, Kanpur, India
pawang.iitk@gmail.com, vipular@iitk.ac.in, lbehera@iitk.ac.in
[2] School of Computing and Intelligent Systems, University of Ulster, UK
l.behera@ulster.ac.uk

Abstract. In this paper, we are presenting our work towards building a dependency parser for Sanskrit language that uses deterministic finite automata(DFA) for morphological analysis and 'utsarga apavaada' approach for relation analysis. A computational grammar based on the framework of Panini is being developed. A linguistic generalization for Verbal and Nominal database has been made and declensions are given the form of DFA. Verbal database for all the class of verbs have been completed for this part. Given a Sanskrit text, the parser identifies the root words and gives the dependency relations based on semantic constraints. The proposed Sanskrit parser is able to create semantic nets for many classes of Sanskrit paragraphs(अनुच्छेद). The parser is taking care of both external and internal sandhi in the Sanskrit words.

1 Introduction

Parsing is the "de-linearization" of linguistic input; that is, the use of grammatical rules and other knowledge sources to determine the functions of words in the input sentence. Getting an efficient and unambiguous parse of natural languages has been a subject of wide interest in the field of artificial intelligence over past 50 years. Instead of providing substantial amount of information manually, there has been a shift towards using Machine Learning algorithms in every possible NLP task. Among the most important elements in this toolkit are state machines, formal rule systems, logic, as well as probability theory and other machine learning tools. These models, in turn, lend themselves to a small number of algorithms from well-known computational paradigms. Among the most important of these are state space search algorithms, [1] and dynamic programming algorithms (Ferro:Souto:Pardo). The need for unambiguous representation has lead to a great effort in stochastic parsing [3].

Most of the research work has been done for English sentences but to transmit the ideas with great precision and mathematical rigor, we need a language that incorporates the features of artificial intelligence. Briggs [4] demonstrated in his article the salient features of Sanskrit language that can make it serve as an Artificial language. Although computational processing of Sanskrit language has been reported in the literature [6] with some computational toolkits [5], and there is work going on towards developing mathematical model and dependency grammar of Sanskrit [7], the proposed Sanskrit parser is being developed for using Sanskrit language as Indian networking language

G. Huet, A. Kulkarni, and P. Scharf (Eds.): Sanskrit CL 2007/2008, LNAI 5402, pp. 200–218, 2009.

(INL). The utility of advanced techniques such as stochastic parsing and machine learning in designing a Sanskrit parser need to be verified.

We have used deterministic finite automata for morphological analysis. We have identified the basic linguistic framework which shall facilitate the effective emergence of Sanskrit as INL. To achieve this goal, a computational grammar has been developed for the processing of Sanskrit language. Sanskrit has a rich system of inflectional endings (vibhakti). The computational grammar described here takes the concept of vibhakti and karaka relations from Panini framework and uses them to get an efficient parse for Sanskrit Text. The grammar is written in 'utsarga apavaada' approach i.e rules are arranged in several layers each layer forming the exception of previous one. We are working towards encoding Paninian grammar to get a robust analysis of Sanskrit sentence. The paninian framework has been successfully applied to Indian languages for dependency grammars [9], where constraint based parsing is used and mapping between karaka and vibhakti is via a TAM (tense, aspect, modality) tabel. We have made rules from Panini grammar for the mapping. Also, finite state automata is used for the analysis instead of finite state transducers. The problem is that the Paninian grammar is generative and it is just not straight forward to invert the grammar to get a Sanskrit analyzer, i.e. its difficult to rely just on Panini sutras to build the analyzer. There will be lot of ambiguities (due to options given in Panini sutras, as well as a single word having multiple analysis). We need therefore a hybrid scheme which should take some statistical methods for the analysis of sentence. Probabilistic approach is currently not integrated within the parser since we don't have a Sanskrit corpus to work with, but we hope that in very near future, we will be able to apply the statistical methods.

The paper is arranged as follows. Section 2 explains in a nutshell the computational processing of any Sanskrit corpus. We have codified the Nominal and Verb forms in Sanskrit in a directly computable form by the computer. Our algorithm for processing these texts and preparing Sanskrit lexicon databases are presented in section 3. The complete parser has been described in section 4. We have discussed here how we are going to do morphological analysis and hence relation analysis. Results have been enumerated in section 5. Discussion, conclusions and future work follow in section 6.

2 A Standard Method for Analyzing Sanskrit Text

The basic framework for analyzing the Sanskrit corpus is discussed in this section. For every word in a given sentence, machine/computer is supposed to identify the word in following structure. $< Word >< Base >< Form >< Relation >$.

The structure contains the root word (<Base>) and its form <attributes of word> and relation with the verb/action or subject of that sentence. This analogy is done so as to completely disambiguate the meaning of word in the context.

2.1 <Word>

Given a sentence, the parser identifies a singular word and processes it using the guidelines laid out in this section. If it is a compound word, then the compound word with सन्धि has to be undone. For example: नदीमागच्छत्=नदीम्+आगच्छत्.

2.2 <Base>

The base is the original, uninflected form of the word. Finite verb forms, other simple words and compound words are each indicated differently. For Simple words: The computer activates the DFA on the ISCII code [8] of the Sanskrit text. For compound words: The computer shows the nesting of internal and external समास using nested parentheses. Undo सन्धि changes between the component words.

2.3 <Form>

The <Form> of a word contains the information regarding declensions for nominals and state for verbs.

- For undeclined words, just write u in this column.
- For nouns, write first.m, f or n to indicate the gender, followed by a number for the case (1 through 7, or 8 for vocative), and s, d or p to indicate singular, dual or plural.
- For adjectives and pronouns, write first a, followed by the indications, as for nouns, of gender (skipping this for pronouns unmarked for gender), case and number.
- For verbs, in one column indicate the class (गण) and voice. Show the class by a number from 1 to 11. Follow this (in the same column) by '1' for parasmaipada, '2' for ātmanepada and '3' for ubhayapada. For finite verb forms, give the root. Then (in the same column) show the tense as given in Table 3. Then show the inflection in the same column, if there is one. For finite forms, show the person and number with the codes given in Table 2. For participles, show the case and number as for nouns.

2.4 <Relation>

The relation between the different words in a sentence is worked out using the information obtained from the analysis done using the guidelines laid out in the previous

Table 1. Codes for <Form>

pa/	passive
ca/	causative
de/	desiderative
fr/	frequentative

Table 2. Codes for Finite Forms, showing the Person and the Number

1	प्रथम पुरुष
2	मध्यम पुरुष
3	उत्तम पुरुष
s	singular
d	dual
p	plural

Table 3. Codes for Finite verb Forms, showing the Tense

pr	present
if	imperfect
iv	imperative
op	optative
ao	aorist
pe	perfect
fu	future
f2	second future
be	benedictive
co	conditional

Table 4. Codes for <Relation>

v	main verb
vs	subordinate verb
s	subject(of the sentence or a subordinate clause)
o	object(of a verb or preposition)
g	destination(gati) of a verb of motion
a	Adjective
n	Noun modifying another in apposition
d	predicate nominative
m	other modifier
p	Preposition
c	Conjunction
u	vocative, with no syntactic connection
q	quoted sentence or phrase
r	definition of a word or phrase(in a commentary)

subsections. First write down a period in this column followed by a number indicating the order of the word in the sentence. The words in each sentence should be numbered sequentially, even when a sentence ends before the end of a text or extends over more than one text. Then, in the same column, indicate the kind of connection the word has to the sentence, using the codes given in table 4. Then, in the same column, give the number of the other word in the sentence to which this word is connected as modifier or otherwise. The relation set given above is not exhaustive. All the 6 karakas are defined as in relation to the verb.

3 Algorithm for Sanskrit Rulebase

In the section to follow in this paper, we shall explain two of the procedures/algorithms that we have developed for the computational analysis of Sanskrit. Combined with these algorithms, we have arrived at the skeletal base upon which many different modules for Sanskrit linguistic analysis such as: relations, सन्धि, समास can be worked out.

3.1 Sanskrit Rule Database

Every natural language must have a representation, which is directly computable. To achieve this we have encoded the grammatical rules and designed the syntactic structure for both the nominal and verbal words in Sanskrit. Let us illustrate this structure for both the nouns and the verbs with an example each.

Noun: Any noun has three genders: Masculine, Feminine and Neuter. So also the noun has three numbers: Singular, Dual and Plural. Again there exists eight classification in each number: Nominative, Accusative, Imperative, Dative, Ablative, Genitive, Locative and Vocative. Interestingly these express nearly all the relations between words in a sentence.

In Sanskrit language, every noun is deflected following a general rule based on the ending alphabet such as अकारान्त. For example, राम is in class अकारान्त which ends

Table 5. Attributes of the declension for noun

$Class^*$		$Case^\eta$	$Gender^\varsigma$
अकारान्त(1)	दकारान्त(14)	कर्त्ता(1)	पुल्लिङ(1)
आकारान्त(2)	धकारान्त(15)	कर्म(2)	स्त्रीलिङ(2)
इकारान्त(3)	नकारान्त(16)	करण(3)	नपुंसकलिङ(3)
ईकारान्त(4)	नकारान्त(17[1])	सम्प्रदान(4)	$Number^@$
उकारान्त(5)	पकारान्त(18)	अपादान(5)	एकवचन(1)
ऊकारान्त(6)	भकारान्त(19)	सम्बन्ध(6)	द्विवचन(2)
ऋकारान्त(7)	रकारान्त(20)	अधिकरण(7)	बहुवचन(3)
ऐकारान्त(8)	वकारान्त(21)	सम्बोधन(8)	
ओकारान्त(9)	शकारान्त(22)		
औकारान्त(10)	षकारान्त(23)		
चकारान्त(11)	सकारान्त(24)		
जकारान्त(12)	हकारान्त(25)		
तकारान्त(13)			

with अ(a). Such classifications are given in Table 5. Each of these have different inflections depending upon which gender they correspond to. Thus अकारान्त has different masculine and neuter declensions, आकारान्त has masculine and feminine declensions, इकारान्त has masculine, feminine and neuter declensions. We have then encoded each of the declensions into ISCII code, so that it can be easily computable in the computer using the algorithm that we have developed for the linguistic analysis of any word.

Let us illustrate this structure for the noun with an example. For अकारान्त, masculine, nominative, singular declension:
This is encoded in the following syntax: $(163\{1^*, 1^\eta, 1^\varsigma, 1^@\})$.

Where 163 is the ISCII code of the declension (Table 6). The four 1's in the curly brackets represent Class, Case, Gender and Number respectively (Table 5).

Table 6. Noun example

अ Masculine	Singular(एकवचन)	
	Endings	ISCII Code
Nominative	:	163

Pronouns: According to Paninian grammar and Kale, [10] Sanskrit has 35 pronouns which are: सर्व, विश्व, उभ, उभय, इतर, इतम, अन्य, अन्यतर, इतर, त्वत्, त्व, नेम, यम, यिम, पूर्व, पर, अवर, दक्षिण, उपर, अधर, स्व, अन्तर, त्यद्, एतद्, इतद्, अदस्, एक, द्वि, युष्मद, भवत् and किम्.

We have classified each of these pronouns into 9 classes: Personal, Demonstrative, Relative, Indefinitive, Correlative, Reciprocal and Possessive. Each of these pronouns have different inflectional forms arising from different declensions of the masculine and feminine form. We have codified the pronouns in a form similar to that of nouns.

Adjectives: Adjectives are dealt in the same manner as nouns. The repetition of the linguistic morphology is avoided.

Verbs: A Verb in a sentence in Sanskrit expresses an action that is enhanced by a set of auxiliaries"; these auxiliaries being the nominals that have been discussed previously.

The meaning of the verb is said to be both *vyapara* (action, activity, cause), and *phala* (fruit, result, effect). Syntactically, its meaning is invariably linked with the meaning of the verb "to do". In our analysis of Verbs, we have found that they are classified into 11 classes(गण, Table 7). While coding the endings, each class is subdivided according to "इट्" knowledge, सेट् , अनिट् and वेट्; each of which is again sub-classified as into 3 sub-classes as आत्मनेपद , परस्मैपद andउभयपद, which we have denoted as pada. Each verb sub-class again has 10 *lakāras*, which is used to express the tense of the action. Again, depending upon the form of the sentence, again a division of form as कर्त्तवाच्य , कर्मवाच्य and भाववाच्य has been done. This classification has been referred to as voice. This structure has been explained in Table 7.

Table 7. Attributes of the declension for verb

$Class^{*}$	it^{γ}	$pada^{\eta}$	$Tense^{\varsigma}$
भ्वादिगण(1)	सेट्(1)	आत्मनेपद(1)	लट्(1)
अदादिगण(2)	अनिट्(2)	परस्मैपद(2)	लङ्(2)
दिवादिगण(3)	वेट्(3)	उभयपद(3)	लृट(3)
स्वादिगण(4)			लोट्(4)
तुदादिगण(5)			विधिलिङ्(5)
रुढादिगण(6)			आर्शीलिङ्(6)
तनादिगण(7)			लिट्(7)
क्रह्यादिगण(8)			लुट्(8)
चुरादिगण(9)			लुङ्(9)
जुहोत्यादिगण(10)			लृङ्(10)
कण्ड्वादिगण(11)			

$Voice^{\lambda}$	$Person^{@}$	$Number^{\delta}$
कर्त्तवाच्य(1)	प्रथम पुरुष(1)	एकवचन(1)
कर्मवाच्य(2)	मध्यम पुरुष(2)	द्विवचन(2)
भाववाच्य(3)	उत्तम पुरुष(3)	बहुवचन(3)

Let us express the structure via an example for भ्वादिगण , परस्मैपद, Present Tense, First person, Singular. This is encoded in the following syntax: $(219(194\{1^{*}, 1^{\gamma}, 2^{\eta}, 1^{\varsigma}, 1^{\lambda}, 1^{@}, 1^{\delta}\}))$.

Where 219194 is the ISCII code of the endings (Table 8). The numbers in curly brackets represent class, "it", pada, tense, voice, person and number respectively (Table 7).

Separate database files for nominals and verbs have been maintained, which can be populated as more and more Sanskrit corpsuses are mined for data. The Sanskrit rule base is prepared using the "Sanskrit Database Maker" developed during this work.

3.2 Deterministic Finite Automata: Sanskrit Rule Base

We have used deterministic finite automata (DFA) [11] to compute the Sanskrit rule base, which we developed as described in section III A. Before we explain the DFA, let us define it.

Table 8. Verb example

PRESENT	Singular(एकवचन)	
	Endings	ISCII Code
First	ति	219194

A deterministic finite automaton consists of:

1. A finite set of states, often denoted Q.
2. A finite set of input symbols, often denoted S.
3. A transition function that takes as arguments a state and input symbol and returns a state, often commonly denoted d.
4. A start state, one of the states in Q, denoted q0.
5. A set of *final* or *accepting states* F. The set F is a subset of Q.

Thus, we can define a DFA in this "five-tuple" notation: $A = (Q, S, d, q0, F)$. With this short discussion of the DFA, we shall proceed to the DFA structure for our Sanskrit Rule Base. Since we are representing any word by ISCII codes that range from 161 to 234, we have effectively 74 input states. In the notation given below, we are representing the character set by $\{C0, C1, \ldots, C73\}$, where Ci is the character corresponding to the ISCII code $161 + i$. Thus, if we define a DFA = $M(Q, S, d, q0, F)$ for our Sanskrit Rule Database, each of the DFA entities are as follows:

- Q = $\{q_0, q_{C0}, q_{C1}, \ldots, q_{C73}\} \times \{0, 1\}$. 0 represents that the state is not a final state and 1 tells that the state is a final state.
- $\sum = \{C0, C1, \ldots, C73\}$
- $\delta((q_x, a), Y) = \delta(q_Y, a)$or $\delta(q_Y, b)$ a,b $\epsilon\{0, 1\}$
- $q_0 =< q_0, 0 >$
- $F \subset \{q_{C0}, q_{C1}, \ldots, q_{C73}\} \times \{1\}$

In this work, we have made our DFA in a matrix form with each row representing the behavior of a particular state. In a given row, there are 74 columns and entries in a particular column of the corresponding row store the state we will finally move to on receiving the particular input corresponding to the column. In addition, each row carries the information whether or not it is a final state.

For example: $D[32][5] = 36$ conveys that in the DFA matrix D[i][j], in 32nd state, if input is $C5$, we will move to state no. 36 .(To be noted: $C5$ is the character corresponding to the ISCII code 166.).

In the graph below, we are giving an example how the DFA will look as a tree structure. The particular graph is constructed for the verb declensions for the class भ्वादिगण. The pada is परस्मैपद and the tense is present tense. The search in this DFA will be as follows:- If the first ending of the input corresponds to one of the state 163, 195 or 219, we will move ahead in the DFA otherwise the input is not found in this tree. On getting a match, the search will continue in the matched branch.

In general, the search in the DFA is done as follows (We take the example of searching for भवथः in the DFA tree constructed above:-

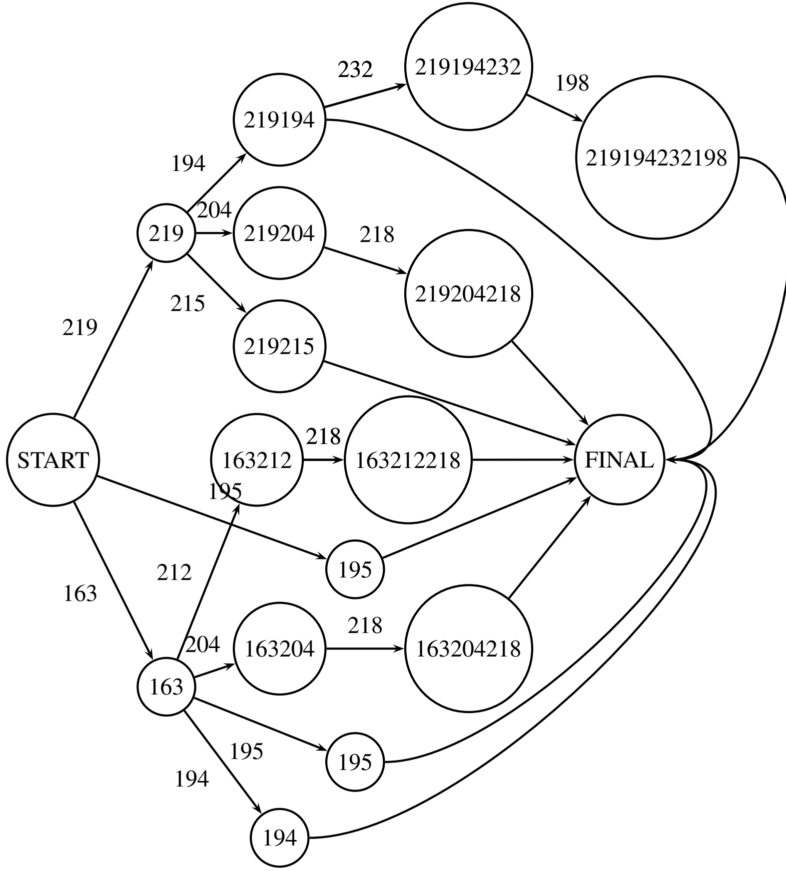


Fig. 1. DFA tree obtained for भ्वादिगण परस्मैपद present tense

- Firstly, an input word is given as the input to the user interface in Devanagari format.
- The word is changed to its equivalent ISCII code (203212195163 in this case).
- The automaton reads the forms in the reverse order to lemmatize them. In our DFA, we give one by one the last three digits of the ISCII code till the matching is there.
  - Start state: 000
  - input to DFA: 163, i.e character $C2$.
  - In the DFA matrix we will check the entry D[0][2]. If it is zero, no match is there for this entry and hence no match either for the word. Else we will move to the state specified by the entry.
  - In this case, we get the entry corresponding to state "163". That means it is either an intermediate state or a final state. From the graph, it is visible that the tree accepts 163 just after the start state. Also, it is not a final state. Now we will have 195 (i.e. $C34$) as next input and 34^{th} column of the row corresponding to state 232 will be checked and the search continues till no match.
  - Final match will be 163195.

- The final match will be checked for being eligible for a final state which is true in this case. We can verify it from the graph given.
- Remaining part of the word is sent to database engine of program to verify and to get attributes. The word corresponding to the stem, Devanagari equivalent of 203212, that is भव) will be sent to database.
- If both criteria are fulfilled (final state match and stem match through database), we will get the root word and its category (verb in this case). The attributes such as tense, form, voice, class, pada, person, number are coded in the final state itself according to the notations given in table 5 and 7. All the possible attributes are stored and it is left up to the final algorithm to come up with the most appropriate solution.

Let me just explain how we have obtained the deterministic finite automata. Clearly, the states are obtained via input symbols. Ambiguity remains in $\{0, 1\}$. If the state is not a final state at all, it is declared as intermediate state without any ambiguity to be considered for non-deterministic. When the state is a final state, for example consider गच्छति and गमिष्यति. When we encounter ति in गच्छति, we get the root as गच्छ. (Of course, we have to add the हलन्त and go through धात्वादेश getting गम् as root verb.) But in गमिष्यति, ति is not a final state. It seems at this point that we could have obtained a non-deterministic finite automaton. We have resolved the problem by accepting the following facts:

1. Final state can be intermediate state too but not the other way round.
2. Our algorithm doesn't stop just as it gets to a final state, it goes to the highest possible match, checks it for being final state and, in case it isn't, it backtracks and stops at the optimal match which satisfies the two criteria as told in the algorithm (final state match and stem match through database).

There might be another ambiguity too, for example, in रामाभ्याम्, आभ्याम् is a final state but it refers to करण, सम्प्रदान, and अपादान karaka. This seems to be non-deterministic. We have avoided this problem by suitably defining the states. Final state represents all possibilities merged in a single state. It is up to the algorithm to come up with the unique solution. There could be situation where longest match is not the right assignment. To deal with this, all other possible solutions are also stacked and are substituted (if needed) when we go for relation analysis. For example, let us take the word सनाभ्याम्. We assume that सना, सनाभि and सनाभ्या are valid root words. Our algorithm will choose सना as root word along with the attributes (3 possibilities here). But the other solutions are also stacked in decreasing order of the match found. It is discussed in the relation analysis, how we deal with this situation.

4 Algorithm for Sanskrit Parser

The parser takes as input a Sanskrit sentence and using the Sanskrit Rule base from the DFA Analyzer, analyzes each word of the sentence and returns the base form of each word along with their attributes. This information is analyzed to get relations among the words in the sentence using If-Then rules and then output a complete dependency

parse. The parser incorporates Panini framework of dependency structure. Due to rich case endings of Sanskrit words, we are using morphological analyzer. To demonstrate the Morphological Analyzer that we have designed for subsequent Sanskrit sentence parsing, the following resources are built:

- Nominals rule database (contains entries for nouns and pronouns declensions)
- Verb rule database (contains entries for 10 classes of verbs)
- Particle database (contains word entries)

Now using these resources, the morphological analyzer, which parses the complete sentences of the text is designed.

4.1 Morphological Analysis

In this step, the Sanskrit sentence is taken as input in Devanagari format and converted into ISCII format. Each word is then analyzed using the DFA Tree that is returned by the above block. Following along any path from start to final of this DFA tree returns us the root word of the word that we wish to analyze, along with its attributes. While evaluating the Sanskrit words in the sentence, we have followed these steps for computation:

1. First, a left-right parsing to separate out the words in the sentence is done.
2. Second, each word is checked against the Sanskrit rules base represented by the DFA trees in the following precedence order: Each word is checked first against the avavya database, next in pronoun, then verb and lastly in the noun tree.

The reason for such a precedence ordering is primarily due to the fact that *avavya* and *pronouns* are limited in number compared to the verbs, and verbs are in-turn limited compared to the infinite number of nouns that exist in Sanskrit.

Sandhi Module. In the analysis, we have done, the main problem was with words having external sandhi. Unless we are able to decompose the word into its constituents, we are unable to get the morph of the word. So, a rulebase sandhi analyzer is developed which works on the following principles.

- Given an input word, it checks at each junction for the possibility of sandhi.
- If it finds the junction, it breaks the word into possible parts and sends the first part in the DFA.
  - If it finds a match, it sends the second part in DFA.
    - If no match, it recursively calls the sandhi module (For the possibility of multiple sandhi in a single word).
    - If match is found, terminates and returns the words.
  - If no match, it goes to the next junction.

The rules for decomposing the words are taken from Panini grammar. The search proceeds entirely backwards on the syllabic string. Emphasis is given on minimum possible breaks of the string, avoiding overgeneration.

Panini grammar has separate sections for vowel sandhi as well as consonant sandhi. Also, there is specification of visarga sandhi. Below, we are describing the simplified rules for undoing sandhi.

Vowel Sandhi: We have considered दीर्घ संधि, वृद्धि संधि, गुण संधि, यण संधि and अयादि संधि in vowels. (पररूप , पूर्वरूप and प्रकृतिभाव are not taken into account yet.)

1. दीर्घ संधि:- If the junction is the मात्रा corresponding to आ, ई, ऊ or ॠ, it is a candidate for दीर्घ संधि. The algorithm for an example word भानूदय is explained.
   - We assume that we don't get any match at the junction आ after भ.
   - The junction ऊ is a candidate for दीर्घ संधि. So the following breaks are made: 1. भानु+ उदय, 2. भानु+ ऊदय, 3. भानू+उदय, 4. भानू+ ऊदय. For each break, the left hand word is first sent to DFA and only if it is a valid word, right word will be sent. In this case, first solution comes to be the correct one.
2. वृद्धि संधि:- In this case, the junction is ऐ, औ. The corresponding break-ups are:
   - ऐ:- (अ or आ) + (ए or ऐ).
   - औ:- (अ or आ) + (ओ or औ).

   The algorithm remains the same as told in previous case.
3. गुण संधि:- In this case, the junction is ए, ओ, अर्, अल्. The corresponding break-ups are:
   - ए:- (अ or आ) + (इ or ई).
   - ओ:- (अ or आ) + (उ or ऊ).
   - अर्:- (अ or आ) + (ऋ or ॠ).
   - अल्:- (अ or आ) + (ऌ or ॡ).

   The algorithm follows the same guidelines.
4. यण संधि:- In this case, the junction is a halanta followed by य, व, र, ल. The corresponding break-ups are:
   - halanta + य:- (इ or ई) + अ.
   - halanta + व:- (उ or ऊ) + अ.
   - halanta + र:- (ऌ or ॡ) + अ.
   - halanta + ल:- (ऋ or ॠ) + अ.

   The algorithm follows the same guidelines.
5. अयादि संधि:- In this case, the junction is अय्, आय्, अव्, आव् followed by any vowel. The corresponding break-ups are:
   - अय् + vowel:- ए + vowel. (same vowel is retained.)
   - आय् + vowel:- ऐ + vowel.
   - अव् + vowel:- ओ + vowel.
   - आव् + vowel:- औ + vowel.

   The algorithm follows the same guidelines.

Consonant Sandhi: For dealing with consonant sandhi, we have defined some groups taking clue from panini grammar such as कु, चु, टु, तु, पु each of which have 5 consonants which are similar in the sense of place of pronunciation. Also, there is a specific significance of first, second, third etc. letter of a specific string. The following ruleset is made:

- Define string s1, with first five entries of टु and 6th entry as ष. Also, define s2, with first five entries of तु and 6th entry as स. The rule says,

  The junction is $a + halanta + c$, and the breakup will be $b + halanta$ and c, where $a, c \epsilon$ s1, $b \epsilon$ s2 and the position of a and b are same in the respective strings.

  For example, in the word रामष्षष्ठः, the junction is ष + halanta + ष. The break-up will be, स +halanta and ष. Hence we get रामस् + षष्ठः.

– Define string s1, with first five entries of चु and 6th entry as श. Also, define s2, with first five entries of तु and 6th entry as स. The rule says,

  The junction is $a + halanta + c$, and the breakup will be $b + halanta$ and c, where $a, c \epsilon$ s1, $b \epsilon$ s2 and the position of a and b are same in the respective strings.

  For example, in the word सज्जन, the junction is ज + halanta + ज. ज is the third character of string $s1$. The break-up will be, द +halanta and ज. Hence we get सद् + जन.
– We have defined strings as घोष and अघोष with अघोष containing first two characters of all the five strings कु, चु, टु, तु, पु as well as श, ष, स. घोष contains all other consonants and all the vowels. The rule says, if we get a junction with $a + halanta + c$, where a,c ϵ घोष, a will be changed to corresponding अघोष while undoing the sandhi. Similarly, other rules are made.
– The vowels are categorized into ह्रस्व and दीर्घ categories. ह्रस्व contains अ, इ, उ, ऋ and दीर्घ contains आ, ई, ऊ, ॠ. If the junction is a + न + halanta + न, where a ϵ ह्रस्व, the break-up will be: a + न + halanta and ϕ, where ϕ denotes null, i.e. other न is removed. For example, तस्मिन्नरण्ये breaks up into तस्मिन् and अरण्ये.

Visarga Sandhi: We have looked at visarga sandhi in a single word. The rules made are as follows:

– The junction is श + halanta + a ϵ चु. The break-up will be ः and a.
– The junction is ष + halanta + a ϵ टु. The break-up will be ः and a.
– The junction is स + halanta + a ϵ तु. The break-up will be ः and a.
– The junction is र + halanta + a ϵ consonant. The break-up will be ः and a.
– The junction is र + halanta + a ϵ vowel. The break-up will be ः and a.

4.2 Relation Analysis

With the root words and the attributes for each word in hand for the previous step, we shall now endeavor to compute the relations among the words in the sentence. Using these relation values we can determine the structure of each of the sentences and thus derive the semantic net, which is the ultimate representation of the meaning of the sentence.

For computing the relations, we have employed a case-based approach i.e., nominals were classified as subject, object, instrument, recipient (beneficiary), point of separation (apaadaana) and location, to the verb based on the value of the case attribute of the word, as explained under noun example in Section 3.1.

The Sanskrit language has a dependency grammar. Hence the karaka based approach is used to obtain a dependency parse tree. There are reasons for going for dependency parse:

1. Sanskrit is free phrase order language. Hence, we need the same parse for a sentence irrespective of phrase order.
2. Once the karaka relations are obtained, it is very easy to get the actual thematic roles of the words in the sentence.

The problem comes when we have many possible karakas for a given word. We need to disambiguate between them. We have developed some If-Then rules for classifying the nouns, pronouns, verb, sub-verbs and adjectives in the sentence. The rules are as follows: First we are looking at the sentences having at least one main verb. Nominal sentences are to be dealt in the similar manner but the description will be given later.

1. If there is a single verb in the sentence, declare it as the main verb.
2. If there are more than one verb,
   (a) The verbs having suffix क्त्वा , ल्यप् , तुमुन् are declared subverbs of the nearest verb in the sentence having no such affix.
   (b) All other verbs are main verbs of the sentence and relations for all other words are given in regard to the first main verb.
3. For the nouns and pronouns, one state may have many possibilities of the cases. These ambiguities are to be resolved. The hand written rules for determining these ambiguities are as follows (Rules are written for nouns. Adjective precede nouns (May not precede too due to free word order nature.) and hence get the same case as nouns. For pronouns, rules are same as that for nouns.):
   (a) Nominative case: The assumption is that there is only one main subject in an active voice sentence. We proceed as follows:
      – All the nouns having nominal case as one of the attributes are listed. (For example, फलम् has both possibilities of being nominative or accusative case.)
      – All those connected by च are grouped together and others are kept separate. We now match each group along the following lines:
         • The number matches with that of the verb(Singular/dual/plural).
         • The root word matches with the person of the verb(i.e root word "अस्मद्" for 3rd person, "युष्मद्" for 2nd person).

      If ambiguity still remains, the one having masculine/feminine as gender is preferred for being in कर्ता karaka and declared as subject of the main verb.

   In passive voice,
      – Nominative case is related to main verb as an object. After grouping and going through the match, the noun is declared as object of main verb.
   (b) Accusative case: Assuming that the disambiguation for nominative case works well, there is no disambiguation left for this case. All those left with accusative case indeed belong to that. The noun is declared as object to nearest sub-verb or main verb.
   (c) Instrumental case: If the sentence is in passive voice, the noun is declared as subject of the main verb.

   For active voice, ambiguity remains if the number is dual. The folowing rules are used:
      – We seek if the indeclinable such as सह, साकम्, सार्धम्, समम् follow the noun. In that case, noun is declared as instrument.
      – If the noun is preceded by time or distance measure or is itself one of these, it is declared as instrument. For example द्वाभ्याम् दिनाभ्याम् नीरोगः जातः, here द्वाभ्याम् is the disambiguating feature.
      – If बधिरः, काणः are following noun, the noun is declared as instrumental.

(d) Dative case: For dative case, disambiguity is with respect to ablative case in terms of dual and plural nnumbers The disambiguating feature used here is main verb. That is, there are certain verbs which prefer dative case and certain verbs prefer ablative. For example:
   – The verbs preferring dative case are क्रुध्, द्रुह्, ईर्ष्य्, असूय्, स्पृह् etc.
   – The verbs preferring ablative case are जुगुप्सा, विराम, प्रमाद, वार etc.

   Initially, we have populated the list using अष्टाध्यायी knowledge as well as some grammar books but this has to be done statistically using corpus analysis.

(e) Ablative case: The ambiguity here is for certain nouns with the genitive case in singular person. The ambiguity resolution proceeds along the following lines:
   – If the noun having ambiguity has a verb next to it, it will be taken as ablative (Noun with genitive case marker is not followed by a verb.)
   – If suffixes तरप्, तमप् are used in the sentence, the noun is declared as ablative.
   – If तुल्य, सदृश are following the noun, it is declared as genitive.
   – Finally, we look for the disambiguating verbs as done in previous case.

(f) Genitive case: The ambiguity is there in dual with respec to locative case. We have used that by default, it will be genitive since we have not encountered any noun with locative case and dual in number.

(g) Locative case: The ambiguities are already resolved.

Only problematic case will be the situation discussed in section 3.2 with the example of सनाभ्याम्. If the algorithm is able to generate a parse taking the longest possible match, we will not go into stacked possibilities, but if the subject disagrres with the verb (blocking), or some other mismatch is found, we will have to go for stacked possibilities.

Thus, we have got the case markings. Relation for nominative and accusative case markings have already been defined. For other case markings,

– Instrumental: related as an instrument to main verb in certain cases (taken from अष्टाध्यायी).
– Dative: related as recipient to main verb in certain cases, but also denotes the purpose.
– Ablative: related as separation point.
– Genitive: this is not considered as karaka since karaka has been defined as one which takes role in getting the action done. Hence it is related to the word following it.
– Locative: related as location to the main verb.

Still, we have not given any relation to adjectives and adverbs. For each adjective, we track the noun it belongs to and give it the same attributes. It is defined as adjective to the noun. The adverbs are related to the verb it belongs as adverb.

Based on these relations, we can obtain a semantic net for the sentence with verb as the root node and the links between all the nodes are made corresponding to relations with the verb and interrelations obtained.

Sanskrit has a large number of sentences which are said to be nominal sentences, i.e. they don't take a verb. In Sanskrit, every simple sentence has a subject and a predicate.

If the predicate is not a finite verb form, but a substantive agreeing with the subject, the sentence is a nominal sentence. In that case, the analysis that we have done above seems not to be used as it is. But in Sanskrit, there is a notion called आक्षेप, that is, if one of the verb or subject is present, other is obtained to a certain degree of definiteness. Take for example, the sentence अहम् कलमेन लिखामि ।. If instead of saying the full sentence, I say अहम् कलमेन, लिखामि is determined as verb. Similarly, if I say कलमेन लिखामि, the subject अहम् is determined. अहम् is a kind of appositive expression to the inflectional ending of the verb लिखामि. We have used this concept for analyzing the nominal sentences. That is, verb is determined from the subject. Mostly, the forms of अस् only are used and relations are defined with respect to that. Although, the analysis done is not exhaustive, some ruleset is built to deal with them. Most of the times, relations in a nominal sentence are indicated by pronouns, adjectives, genitive. For example, in the sentence सः सुन्दरः बालकः, there is आक्षेप of the verb अस्ति in the sentence by the subject बालकः. Hence बालकः is related to the verb as subject. सः is a pronoun referring to बालकः and सुन्दरः is an adjective referring to बालकः. Similarly, इदम् तस्य गृहम् ।. In this sentence, इदम् is a pronoun referring to गृहम् and तस्य is a genitive to गृहम्. Here again, there will be आक्षेप of the verb अस्ति and गृहम् will be related to the verb as subject.

5 Results

5.1 Databases Developed

The following Sanskrit Rule Databases have been developed during the project:-

- Nominals (शब्दरूप) rule database contains entries for nouns and pronouns declensions along with their attributes.
- Verb (धातुरूप) rule database contains entries for 10 classes of verb along with their tenses.
- Particle (अव्यय) database.

5.2 Parser Outputs

Currently, our parser is giving an efficient and accurate parse of Sanskrit text. Samples of four of the paragraphs which have correctly been parsed are given below along with snapshot of one sentences per paragraph.

अनुच्छेद 1:- पूज्याः गुरुचरणाः रमणीये गुरुकुले प्रातः नियमेन सन्ध्याम् उपास्य मेधाविनः शिष्यान् सम्यक् अध्यापयन्ति। गुरुकुले अनेकेछ्हराः सन्ति। केचन बालाः। केचन प्रौढाः। गुरुचरणाः प्रौढान् छात्रान् पाठयन्ति। प्रौढाः बालान् पाठयन्ति। सर्वे वेदमन्त्रान् विशुद्धरूपेण उच्चारयन्ति। गुरुचरणाः शिष्यान् वेदस्य अर्थम् अपि बोधयन्ति। प्रतिदिनम् प्रातः सायम् सर्वे शिष्याः यज्ञम् अपि कुर्वन्ति। ते वनम् गत्वा समिधः आनयन्ति। ते नियमेन व्यायामम् कुर्वन्ति। एते शिष्याः एव धर्मस्य रक्षणम् करिष्यन्ति ।.

अनुच्छेद 2:- मम गृहे एकम् उपवनम् वर्तते। उपवनस्य समीपे एका गोशाला वर्तते। अहम् प्रतिदिनम् प्रातः सूर्योदयात् पूर्वम् शय्याम् त्यक्त्वा शौचादिकम् विधाय गोशालाम् गच्छामि। तत्र मम गौः माम् प्रतीक्षते। मम गोः नाम धवला अस्ति। धवलायाः वत्सायाः नाम गौरी अस्ति। अहम्

S.no.	Word	Root	Group	Attribute	Relation
1	पूज्याः	पूज्य	Adjective	m8p m1p	1 adjective 2
2	गुरुचरणाः	गुरुचरण	Noun	m8p m1p	2 subject 12
3	रमणीये	रमणीय	Adjective	n8d n7s n2d n	3 adjective 4
4	गुरुकुले	गुरुकुल	Noun	n8d n7s n2d n	4 location 12
5	प्रातः	प्रातः	Avyaya		5 adverb 8
6	नियमेन	नियमेन	Avyaya		6 adverb 8
7	सन्ध्याम्	सन्ध्या	Noun	f2s	7 object 8
8	उपास्य	आस्	sub-verb	उप,क्त्वा	8 sub-verb 12
9	मेधाविनः	मेधाविन्	Adjective	m8p m6s m5s	9 adjective 10
10	शिष्यान्	शिष्य	Noun	m2p	10 object 12
11	सम्यक्	सम्यक्	Avyaya		11 adverb 12
12	अध्यापयन्ति	इंङ	Verb	pr1p pr1p pr1p	12 verb 12

Fig. 2. Parser output for पूज्याः गुरुचरणाः रमणीये गुरुकुले प्रातः नियमेन सन्ध्याम् उपास्य मेधाविनः शिष्यान् सम्यक् अध्यापयन्ति

S.no.	Word	Root	Group	Attribute	Relation
1	अहम्	अस्मद्	Pronoun	g1s	1 subject 11
2	प्रतिदिनम्	प्रतिदिनम्	Avyaya		2 adverb 7
3	प्रातः	प्रातः	Avyaya		3 adverb 7
4	सूर्योदयात्	सूर्योदय	Noun	m5s	4 separation 5
5	पूर्वम्	पूर्व	Pronoun	n2s m2s	5 object 7
6	शय्याम्	शय्या	Noun	f2s	6 object 7
7	त्यक्त्वा	त्यज्	sub-verb	N,क्त्वा	7 sub-verb 11
8	शौचादिकम्	शौचादिक	Noun	n2s n1s	8 object 9
9	विधाय	धा	sub-verb	वि,क्त्वा	9 sub-verb 11
10	गोशालाम्	गोशाला	Noun	f2s	10 object 11
11	गच्छामि	गम्	Verb	pr3s pr3s pr3s	11 verb 11

Fig. 3. Parser output for अहम् प्रतिदिनम् प्रातः सूर्योदयात् पूर्वम् शय्याम् त्यक्त्वा शौचादिकम् विधाय गोशालाम् गच्छामि

धवलाम् प्रेम्णा स्पृशामि। तस्याः सास्नायाम् कण्डूये। सा प्रसन्ना भवति। गौरीम् अपि कराभ्याम् संवाहयामि। तत्पश्चात् गोशालाम् मार्जयामि। गोमयम्, भुक्तावशिष्टानि तृणानि च निरस्यामि। ततः मम धवलायै सुस्वादूनि कोमलानि तृणानि ददे। अनन्तरम् गौरीम् मुञ्चामि। सा झटिति स्वमातरम् धवलाम् धयते। धवलायाः ऊधसि क्षीरम् अवतरति। तत्पश्चात् अहम् धवलाम् दुग्धम् दुहे। एतत् मम नित्यकर्मः। गोसेवाकाले अहम् गोमहिमप्रतिपादकान् वेदमन्त्रान् मन्दम् मन्दम् मधुरस्वरेण उच्चारयामि।

अनुच्छेद 3:- मम गृहे एकम् उपवनम् वर्तते। उपवनम् विशालम् अस्ति। अस्मिन् उपवने नाना फलवृक्षाः सन्ति। बहवः पुष्पतरवः अपि वर्तन्ते। अनेकाः लताः अपि विद्यन्ते। आयुर्वैदिक वनस्पतयः अपि सन्ति। अहम् प्रतिदिनम् सायंकाले उद्यानकर्म कुर्वे। अहम् हानिकराणि तृणानि निराकरोमि।अहम् वारिणा सर्वान् वनस्पतीन् सिञ्चामि।सायंकाले उपवने मम पितृचरणाः भ्रमन्ति। तैः साकम् मम भ्रातृजाः अपि भ्रमन्ति। कदाचित् गौरी अपि तत्र आयाति। सः इतस्ततः वेगेन धावति। शाद्वले हरिततृणानि चरति। मयूराः अपि तत्र आगत्य नृत्यन्ति। ते मधुरम् ध्वनिम् कुर्वते। वृक्षेषु वानराः क्रीडन्ति। विविधाः पक्षिणः वृक्षेषु कूजन्ति। मम मातृचरणाः सर्वेभ्यः रोटिकाखण्डानि धान्यकणान् च वितरन्ति। इदम् दृश्यम् अति आनन्दप्रदम् भवति।

S.no.	Word	Root	Group	Attribute	Relation
1	मम	अस्मद्	Pronoun	g6s	1 relation 2
2	मातृचरणाः	मातृचरणा	Noun	f8p f2p f1p	2 subject 7
3	सर्वेभ्यः	सर्व	Pronoun	n5p n4p m5p r	3 recipent 7
4	रोटिकाखण्डानि	रोटिकाखण्ड	Noun	n8p n2p n1p	4 object 7
5	धान्यकणान्	धान्यकण	Noun	m2p	5 object 7
6	च	च	Avyaya		6 adverb 7
7	वितरन्ति	तृ	Verb	pr1p pr1p pr1p	7 verb 7

Fig. 4. Parser output for मम मातृचरणाः सर्वेभ्यः रोटिकाखन्डानि धान्यकणान् च वितरन्ति

S.no.	Word	Root	Group	Attribute	Relation
1	वेदेषु	वेद	Noun	m7p	1 location 9
2	प्राणिनाम्	प्राणिन्	Noun	m6p	2 relation 5
3	मनुष्याणाम्	मनुष्य	Noun	m6p	3 relation 5
4	च	च	Avyaya		4 adverb 9
5	जीवनाय	जीवन	Noun	m4s	5 recipient 7
6	आवश्यकाः	आवश्यक	Adjective	m8p m1p	6 adjective 7
7	नियमाः	नियम	Noun	m8p m1p	7 subject 9
8	वर्णिताः	वर्णित	Adjective	m8p m1p	8 adjective 7
9	सन्ति	अस्	Verb	pr1p pr1p pr1p	9 verb 9

Fig. 5. Parser output for वेदेषु प्राणिनाम् मनुष्याणाञ्च जीवनाय आवश्यकाः नियमाः वर्णिताः सन्ति

अनुच्छेद 4:- मम देशस्य नाम भारतवर्षम्। मम देशे वेदानाम् अध्ययनम् भवति। वेदेषु प्राणिनाम् मनुष्याणाञ्च जीवनाय आवश्यकाः नियमाः वर्णिताः सन्ति। वेदेषु अन्यतमः ऋग्वेदः। ऋग्वेदे एकम् वाक्यम् वर्तते,"हे मानव! त्वम् कृषिम् अवश्यम् कुरु।" सम्पूर्णे जगति जन्तूनाम् जीवनाधारः अन्नमेव अन्नञ्च कृषिम् विना नैव सम्भवति। अतः भारतीयाः आचार्याः सर्वेषाम् जीवानाम् कल्याणाय कृषिकर्मशिक्षणाय बहुविधान् उपायान् अशिक्षयन्त। तेषु प्रधानः उपायः गोवंशसेवा। गोः वंशे धेनवः, बलीवर्दाः वृषभाः, वत्साः, वृद्धाः धेनवः सर्वे गोवंशजाः अन्तर्भवन्ति। भूमेः बलवर्धनाय गोमूत्रम् गोमयम् च अत्यन्तम् उपकुरूतः। बलदाः भूमिम् कर्षितुम् उपयुज्यन्ते। शस्यानाम् नयनानयनाभ्याम् गोमयजन्यउर्वरकाणाम् इतरपदार्थानाम् इतस्ततः प्रापणाय शकटानाम् प्रयोगः आवश्यकः। शकटाश्च वृषभैरेव उह्यन्ते। अतः गोवंशस्य कृषेश्च अविच्छेद्य कश्चन अपूर्वः सम्बन्धः वर्तते। अतएव गोः उपमा दातुम् न शक्यते। भूमेः एकम् नाम गौः इत्यपि विद्यते। संस्कृते गौः इति पदेन स्त्री गौ पुङ्गवश्च उभावपि गृह्यते। कृषिवर्धनाय शस्यनाशकानाम् कीटानाम् नाशार्थम् केचन घातकद्रव्यानाम् प्रयोगम् कुर्वते। सामवेदस्य प्रथमे आर्चिके कृषिविषये गम्भीरा चर्चा प्राप्यते। अस्माकम् गुरवः महर्षयः सम्पूर्ण जगतः कल्याणाय कृषिविद्याम् सम्यक् प्रचारयन्। कृषिकर्मणे जनान् प्रेरितवन्तः। वेदेषु भूमिः माता इति कथ्यते। यथा माता अस्मान् पालयति तथा पृथ्वी सर्वान् जीवान् अन्नेन फलैः नानाविधपदार्थैः पालयति। यन्त्रैः, तैलेन्धनेन, रसायनद्रव्यैः कीटनाशकैः च या कृषि क्रियते तेन सर्वेषाम् जीवानाम् नाश एव भवति। अनेन कर्मणा सर्वेषाम् प्राणिनाम् मनुष्याणाञ्च महती हानि जायते।

The parse results pave the way for representing the sentence in the form of a Semantic Net. We here give the semantic net for the parse output given in Figure 2.The Semantic Net is shown in Figure 6.

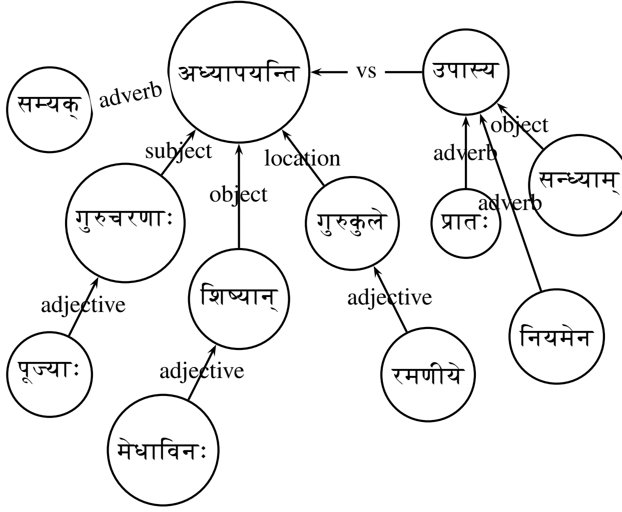


Fig. 6. Semantic net representation of the sentence [पूज्याः गुरुचरणाः रमणीये गुरुकुले प्रातः नियमेन सन्ध्याम् उपास्य मेधाविनः शिष्यान् सम्यक् अध्यापयन्ति]

6 Conclusions and Future Work

Our parser has three parts. First part takes care of the morphology. For each word in the input sentence, a dictionary or a lexicon is to be looked up, and associated grammatical inforation is retrieved. One of the criterion to judge a morphological analyzer is its speed. We have made a linguistic generalization and declensions are given the form of DFA, thereby increasing the speed of parser. Second part of the parser deals with making "Local Word Groups". As noted by Patanjali, any practical and comprehensive grammar should be written in 'utsarga apavāda' approach. In this approach rules are arranged in several layers each forming an exception of the previous layer. We have used the 'utsarga apavāda' approach such that conflicts are potentially taken care of by declaring exceptions. Finally, words are grouped together yielding a complete parse. The significant aspect of our approach is that we do not try to get the full semantics immediately, rather it is extracted in stages depending on when it is most appropriate to do so. The results we have got are quite encouraging and we hope to analyze any Sanskrit text unambiguously.

To this end, we have successfully demonstrated the parsing of a Sanskrit Corpus employing techniques designed and developed in section 2 and 3. Our analysis of the Sanskrit sentences in the form of morphological analysis and relation analysis is based on sentences as shown in the four paragraphs in previous section. The algorithm for analyzing compound words is tested separately. Hence future works in this direction include parsing of compound sentences and incorporating stochatic parsing. We need to take into account the नामधातु as well. We are trying to come up with a good enough lexicon so that we can work in the direction of समास विच्छेद in Sanskrit sentences. Also, we are working on giving all the rules of Panini the shape of multiple layers. In

fact, many of the rules are unimplementable because they deal with intentions, desires etc. For that, we need to build an ontology schema. The Sandhi analysis is not complete and some exceptional rules are not coded. Also, not all the derivational morphology is taken care of. We have left out many प्रत्यय. Reason behind not incorporating the प्रत्यय was that it is difficult to come up with a general DFA tree for any of the प्रत्यय because of the wide number of rules applicable. For that, we need to encode the Panini grammar first.

Acknowledgment

We humbly acknowledge our gratitude to revered Aacharya Sanskritananda Hari, founder and director of Kaushalya pitham Gurukulam, Vadodara for educating us in all aspects of Sanskrit language.

References

1. Bonet, B., Geffner, H.: Planning as heuristic search. Artificial Intelligence 129 (2001)
2. Ferro, M.V., Souto, D.C., Pardo, M.A.A.: Dynamic programming as frame for efficient parsing. Computer science (1998)
3. Ivanov, Y.A., Bobick, A.F.: Recognition of visual activities and interactions by stochastic parsing. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(8), 852–872 (2000)
4. Briggs, R.: Knowledge Representation in Sanskrit and artificial Intelligence. The AI Magazine, 33–39 (1985)
5. Huet, G.: The Zen Computational Linguistics Toolkit. ESSLLI 2002, Lectures, Trento, Italy (2002)
6. Huet, G.: A Functional Toolkit for Morphological and Phonological Processing, Application to a Sanskrit Tagger. Journal of Functional Programming 15(4), 573–614 (2005)
7. Huet, G.: Shallow syntax analysis in Sanskrit guided by semantic nets constraints. In: Proceedings of the 2006 International Workshop on Research Issues in Digital Libraries, Kolkata, West Bengal, India. ACM, New York (2007), yquem.inria.fr/~huet/PUBLIC/IWRIDL.pdf
8. Bureau of Indian Standards. ISCII: Indian Script Code for Information Interchange. ISCII-91(1999)
9. Bharati, A., Sangal, R.: Parsing Free Word Order Languages in the Paninian Framework. In: Proc. of Annual Meeting of Association for Computational Linguistics, New Jersey, pp. 105–111 (1993)
10. Kale, M.R.: A Higher Sanskrit Grammar, 4th edn. Motilal Banarasidass Publishers Pvt. Ltd. (1967)
11. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages and Computation, 2nd edn. Pearson Education Pvt. Ltd., London (2002)

Inflectional Morphology Analyzer for Sanskrit

Girish Nath Jha, Muktanand Agrawal, Subash, Sudhir K. Mishra, Diwakar Mani, Diwakar Mishra, Manji Bhadra, and Surjit K. Singh

Special Centre for Sanskrit Studies
Jawaharlal Nehru University
New Delhi-110067
girishj@mail.jnu.ac.in

The paper describes a Sanskrit morphological analyzer that identifies and analyzes inflected noun-forms and verb-forms in any given sandhi-free text. The system which has been developed as java servlet RDBMS can be tested at http://sanskrit.jnu.ac.in (Language Processing Tools > Sanskrit Tinanta Analyzer/Subanta Analyzer) with Sanskrit data in Unicode text. Subsequently, the separate systems of subanta and tiṅanta will be combined into a single system of sentence analysis with karaka interpretation. Currently, the system checks and labels each word as three basic POS categories - subanta, tiṅanta, and avyaya. Thereafter, each subanta is sent for subanta processing based on an example database and a rule database. The verbs are examined based on a database of verb roots and forms as well by reverse morphology based on Paninian techniques. Future enhancements include plugging in the amarakośa (http://sanskrit.jnu.ac.in/amara) and other noun lexicons with the subanta system. The tiṅanta will be enhanced by the kṛdanta analysis module being developed separately.

Keywords: morphology, analyzer, subanta, tiṅanta, kṛdanta, taddhita, strī-pratyaya, samāsa, avyaya, kāraka, vibhakti, vacana, sandhi, pada, prātipadika, pratyaya, sup, tiṅ, Pāṇini, sūtra, Aṣṭādhyāyī, POS, dhātu, dhātupāṭha, gaṇa, gaṇapāṭha, lakāra, dhāturūpa, śabdarūpa, java, JSP, servlet, Apache-Tomcat, RDBMS, SQL server, JDBC, Unicode.

1 Introduction

The authors in the present paper are describing the morph analyzer (*subanta* and *tiṅanta* analysis systems) for Sanskrit currently running as separate modules at http://sanskrit.jnu.ac.in. Sanskrit is a heavily inflected language, and depends on nominal and verbal inflections for communication of meaning. A fully inflected unit is called *pada*. The *subanta padas* are the inflected nouns and the *tiṅanta padas* are the inflected verbs. Hence identifying and analyzing these inflections are critical to any further processing of Sanskrit.

The results from the *subanta* analyzer for the input text fragment

आम्रस्य आत्मकथा

चपलाः बालकाः आम्राणाम् उद्यानं गच्छन्ति । तत्र आम्रफलानि पश्यन्ति प्रसन्नाः च भवन्ति ।

G. Huet, A. Kulkarni, and P. Scharf (Eds.): Sanskrit CL 2007/2008, LNAI 5402, pp. 219–238, 2009.

are displayed as follows –

आम्रस्य [आम्र+ङस्, षष्ठी, एकवचन] आत्मकथा [आत्मकथा (स्त्रीलिङ्ग) + सु, प्रथमा, एकवचन] <*_PUNCT> चपलाः [चपल (पुल्लिङ्ग) + जस्, प्रथमा, बहुवचन] बालकाः [बालक (पुल्लिङ्ग) + जस्, प्रथमा, बहुवचन] आम्राणाम् [आम्र (पुल्लिङ्ग) + आम्, षष्ठी, बहुवचन] उद्यानं [उद्यन्+अम्, द्वितीया, एकवचन] <गच्छन्ति_VERB> <।_PUNCT> <तत्र_AV> आम्रफलानि [आम्रफल+जस्/शस् प्रथमा/द्वितीया, बहुवचन] <पश्यन्ति_VERB> प्रसन्नाः [प्रसन्न (पुल्लिङ्ग) + जस्, प्रथमा, बहुवचन] <च_AV> <भवन्ति_VERB> <।_PUNCT>

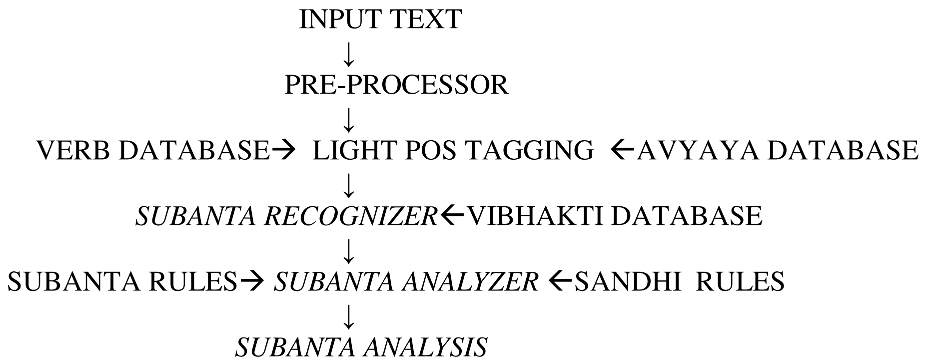
Those in single angled brackets (‘< >’) are non *subanta* categories and those in double angled brackets (‘<< >>’) are possible errors. Since the system does not report any error in this sample, there are no ‘<< >>’ labeled items here. Others are the *subanta* padas as analyzed by the system.

The word गच्छन्ति from the above input text resulted in the following output from the *tiṅanta* analyzer system –

गच्छन्ति { (कर्तृवाच्य) गम् ([भ्वादिगण] [अनिट्] [सकर्मक]) ([लट्]) झि ([परस्मै] [प्रथम–पुरुष] [बहुवचन]) }

2 The *Subanta* Anlyzer

The system accepts Unicode (UTF-8) sandhi free Devanāgarī Sanskrit inputs (word, sentence or text) and processes it according to the following sequence -



The PREPROCESSOR does the simplification and normalization of the Sanskrit text (for example, deletes Roman characters, other invalid words, punctuations etc). The POS TAGGER identifies four categories AVyaya, VERB, PUNCTuation and SUBANTA. The SUBANTA RECOGNIZER does *vibhakti* identification and isolation by searching the *vibhakti* database. The SUBANTA ANALYZER does analysis by checking the *subanta* rule base and sandhi rules. Analysis includes splitting the NPs into its constituents - base [(*prātipadika*) (PDK)], case-number markers (*kāraka-vacana-vibhakti*).

3 Sanskrit Sentence and Basic POS Categories

A Sanskrit sentence has NPs (including AVs), and VPs. Cordona[1] (1988) defines a sentence as -

$$(N - E^n)\ p\ .\ .\ .\ (V - E^v)\ p$$

After *sup* and *tiṅ* combine with PDK, they are assigned syntactico-semantic relation by the *kāraka* stipulations to return complete sentences.

3.1 Sanskrit *Subanta* (Inflected Nouns)

Sanskrit nouns are inflected with seven case markers in three numbers. Potentially, a noun can be declined in all three genders. Sanskrit noun forms can be further complicated by being a derived noun as primary (*kṛdanta*), secondary (*taddhitānta*), feminine forms (*strīpratyayānta)* and compounds (*samāsa).* They can also include *upasargas* and AVs etc. According to Pāṇini, there are 21 case suffixes called *sup* (seven *vibhaktis* combined with three numbers)[2], which can attach to the nominal bases (PDK) according to the syntactic category, gender and end-character of the base. Pāṇini has listed these as sets of three as:

su, au, jas
am, auṭ, śas
ṭā, bhyām, bhis
ṅe, bhyām, bhyas
ṅasi, bhyām, bhyas
ṅas, os, ām
ṅi, os, sup[3]

for singular, dual and plural[4] respectively. These suffixes are added to the PDKs[5] (any meaningful form of a word, which is neither a root nor a suffix) to obtain inflected forms NPs. PDKs are of two types: primitive and derived. The primitive bases are stored in g*aṇapāṭha* [(GP) (collection of bases with similar forms)] while the latter are formed by adding the derivational suffixes. NPs are of mainly six types –

3.1.1 *Avyaya Subanta* (Indeclinable Nouns)

Avyaya subanta, remain unchanged under all morphological conditions[6]. According to Pāṇini [2.2.82][7], affixes *cāp, ṭāp, ḍāp,* (feminine suffixes) and *sup* are deleted by *luk*

---

1 George Cardona, 1988 Pāṇini, His Work and its Traditions, vol **...** i (Delhi: MLBD, 1988).

2 स्वौजसमौट्छष्टाभ्याम्भिस्ङेभ्याम्भ्यस्ङसिभ्याम्भ्यस्ङसोसांङ्योस्सुप्

3 सुपः

4 द्व्येकयोर्द्विवचनैकवचने

5 अर्थवधातुरप्रत्ययः प्रातिपदिकम् ।१।२।४५।।, कृत्तद्धितसमासाश्च ।१।२।४६।।

6 सदृशं त्रिषु लिङ्गेषु सर्वासु च विभक्तिषु ।
वचनेषु च सर्वेषु यन्न व्येति तदव्ययम् ।। [गोपथ ब्राह्मण]

7 अव्ययादाप्सुपः [२.४.८२]

when they occur after an AVs. Pāṇini defines AVs as *svarādinipātamavyayam* [1.1.36], *kṛnmejantaḥ* [1.1.38], *ktvā tosun kasunaḥ* [1.139] and *avyayībhāvaśca* [1.1.40][8] etc.

3.1.2 Basic *Subantas* (Primitive Nouns)

Basic *subantas* are formed by primitive PDKs found in the Panini's *gaṇapāṭha*. For our purpose, all those nouns, the base or inflected form of which can be found in a lexicon can be considered basic *subantas*. Sometimes, commonly occurring primary or secondary derived nouns, feminine or compound forms can also be found in the lexicon. Therefore such *subantas* are also considered basic and do not require any reverse derivational analysis unless specifically required. Such inflected nouns are formed by inflecting the base or PDKs (*arthavadadhāturapratyayaḥ prātipadiakam)* with *sup*. For example: *rāmaḥ, śyāmaḥ, pustakālayaḥ, vidyālayaḥ* etc.

3.1.3 *SamāSāNta Subanta* (Compound Nouns)

Simple words (*padas*), whether substantives, adjectives, verbs or indeclinables, when added with other nouns, form *samāsa* (compound). Sanskrit *samāsas* are divided into four categories, some of which are divided into sub-categories. The four main categories of compounds are as follows:

- adverbial or *avyayībhāva*,
- determinative or *tatpuruṣa,*
- attributive or *bahuvrīhi* and
- copulative or *dvandva*. *dvandva* and *tatpuruṣa* compounds may be further divided into sub-categories

3.1.4 *KṛDanta Subanta* (Primary Derived Nouns)

The primary affixes called *kṛt* are added to verbs to derive substantives, adjectives or indeclinables.

3.1.5 *TaddhitāNta Subanta* (Secondary Derived Nouns)

The secondary derivative affixes called *taddhita* derive secondary nouns from primary nouns. For example - *dāśarathī, gauṇa* etc.

3.1.6 *StrīPratyayāNta Subanta* (Feminine Derived Nouns)

Sanskrit has eight feminine suffixes *ṭāp, cāp ḍāp, ṅīṣ, ṅīn, ṅīp, uṅ* and *ti* etc. and the words ending in these suffixes are called *strīpratyayānta* For example - *ajā, gaurī, mūṣikā, indrāṇī, gopī, aṣṭādhyāyī, kurucarī, yuvatī, karabhorū* etc.

4 Recognition of Sanskrit *subanta*

4.1 Recognition of Punctuations

System recognizes punctuations and tags them with the label PUNCT. If the input has any extraneous characters, then the input word will be cleaned from these elements

[8] स्वरादिनिपातमव्ययम् [१.१.३६], कृन्मेजन्तः [१.१.३८], क्त्वा–तोसुन्–कसुनः [१.१.३९], अव्ययीभावश्च [१.१.४०]

(i.e. 'normalized') so that only Devanāgarī Sanskrit input text is sent to the analyzer. For example, "रा/&^%@#म:, बा,'":–=लक:" ➔ रामः, बालकः

4.2 Recognition of *Avyayas*

System takes the help of *avyaya* database for recognizing AVs. If an input word is found in the AVs database, it is labeled AV, and excluded from the *subanta* analysis as AVs do not change forms after *subanta* affixation. We have stored most AVs in the *avyaya* database.

4.3 Recognition of Verbs

System takes the help of verb database for verb recognition. If an input is found in the verb database, it is labeled VERB and thus excluded from *subanta* analysis. Since storing all Sanskrit verb forms is not possible, we have stored verb forms of commonly used 450 verb roots.

4.4 Recognition of *subanta*

Thus, a process of exclusion identifies the nouns in a Sanskrit text. After the punctuations, *avyayas* and verbs are identified, the remaining words in the text are labeled SUBANTA.

5 Analysis of *subanta*

System does analysis of inflected nouns with the help of two relational database - examples and rules. Brief description of these databases follows-

5.1 Example Database

All complicated forms (which are not analyzed according to any rule) including those of some pronoun are stored the database. For example: अहम्=अस्मद+सु प्रथमा एकवचन;अहं=अस्मद+सु प्रथमा एकवचन;आवाम्=अस्मद+औ प्रथमा द्वितीया द्विवचन;आवां=अस्मद+औ प्रथमा द्वितीया द्विवचन;वयम्=अस्मद+जस प्रथमा बहुवचन;वयं=अस्मद+जस प्रथमा बहुवचन;माम्=अस्मद+अम द्वितीया एकवचन;मां=अस्मद+अम द्वितीया एकवचन

5.2 Rule Database

The *subanta* patterns are stored in this database. This database analyzes those nouns which match a particular pattern from the rule base. For example, रामः, नदी, रमा, पुस्तकम etc. First, the system recognizes *vibhakti* as the end character of nouns. For example, '**:**' is found in nominative singular (1-1) like -रामः, श्यामः, सर्वः, भरतः एकः . The system isolates '**:**' and searches for analysis in the *sup* rule base. In the case of

nominative and accusative dual (1-2/2-2), PDK forms will be 'ौ' ending, for example - रामौ, श्यामौ, सर्वौ, एकौ. The system isolates 'ौ' and searches for analysis by matching in the rule database. The sample data is as follows –

ा=ा+सु प्रथमा एकवचन;ाभ्याम्=+भ्याम् तृतीया चतुर्थी पञ्चमी द्विवचन;ाभ्यां=+भ्याम् तृतीया चतुर्थी पञ्चमी द्विवचन;भ्याम्=+भ्याम् तृतीया चतुर्थी पञ्चमी द्विवचन;भ्यां=+भ्याम् तृतीया चतुर्थी पञ्चमी द्विवचन;ेभ्यः=+भ्यस् चतुर्थी पञ्चमी बहुवचन;भ्यः=+भ्यस् चतुर्थी पञ्चमी बहुवचन;

5.3 Verb Data Sample

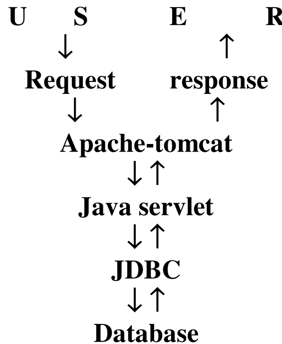
भवति,भवतः,भवन्ति,भवसि,भवथः,भवथ,भवामि,भवावः,भवामः,भवतु,भवताम्,भवन्तु,भव,भवतम्,भवत, भवानि,भवाव,भवाम,अभवत्,अभवताम्,अभवन्,अभवः,अभवतम्,अभवत,अभवम्,अभवाव,अभवाम,भव ेत्,भवेताम्,भवेयुः,भवेः,भवेतम्,भवेत,भवेयम्,भवेव,भवेम,बभूव,बभूवतुः,बभूवुः,बभूविथ,बभूवथुः,बभूव, बभूव,बभूविव

5.4 *Avyaya* Data Sample

अ,कश्चित्,सदैव,अकस्मात्,अकाण्डे,अग्निसात्,अग्नी,अघोः,अङ्ग,अजस्रम्,अञ्जसा,अतः,अति,अतीव, अत्र,अथ,अथकिम्,अथवा,अथो,अद्धा,अद्य,अद्यापि,अधरात्,अधरेद्युः,अधरेण,अधः,अधस्तात्,अधि,अधि हरि,अधुना,अधोऽधः,अध्ययनतः,अनिशम्,अनु,अनेकधा,अनेकशः,अन्तः,अन्तरा,अन्तरेण,अन्यतः,अन्यत्, अन्यत्र

5.5 Architecture of the System

The following model describes the interaction between multi-tiered architecture of the *subanta* analyzer:



5.6 Front-End: Online Interface

The Graphical User Interface (GUI) is produced by JSP (Java Server Pages). The JSP interface allows the user to give input in Devanagari utf-8 format using HTML text area component. The user interface is displayed as follows:

Address http://sanskrit.jnu.ac.in/subanta/rsubanta.jsp

Computational Linguistics R&D
Special Centre for Sanskrit Studies
Jawaharlal Nehru University
New Delhi

Home | Language Processing Tools | Lexical Resources | e-Learning | Corpora/e-Text | Dissertation | Feedback

सङ्गणक द्वारा सुबन्त पहचान और प्रकृति-प्रत्यय विभाग
(Sanskrit Subanta Recognizer and Analyzer)

How does it work   Limitations of this work

The "Sanskrit Subanta Recognizer and Analyzer" is a result of the research carried out by Subhash Chandra (M.Phil. 2004-2006) under the supervision of Dr. Girish Nath Jha for the award of M.Phil. degree. The coding for the application was done by Dr. Girish Nath Jha.

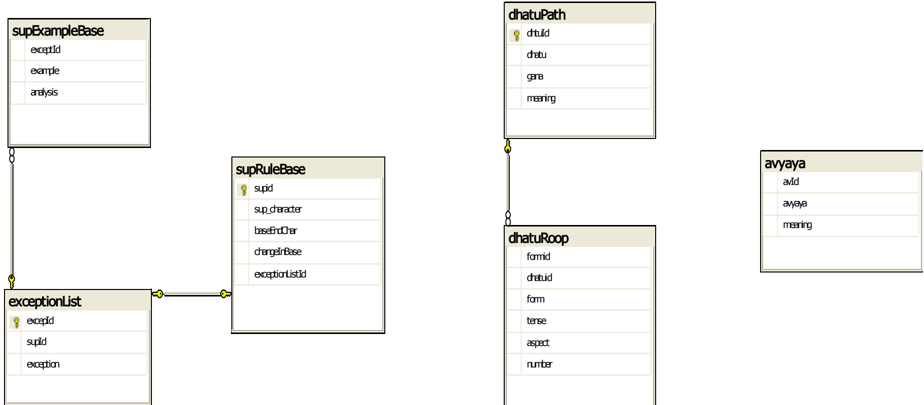
सुबन्त पहचान एवं प्रकृति-प्रत्यय विभाग के लिये कृपया संस्कृत पद, वाक्य या गद्य लिखें

(Devanagari unicode only) cut & paste test data from here

प्रकृति-प्रत्यय विभाग के लिये यहाँ क्लिक करें

5.7 Back-End: Database / Txt Files

There are two versions of the system; the server-based version connects to a MSSQL Server 2005 RDBMS through JDBC. The rule base, example base and other linguistic resources are stored as Devanagari utf-8. The PC based portable version, for obvious reasons, cannot have RDBMS support. Therefore, we have our rules and data stored in utf-8 text files as backend. A design of the reverse *subanta* database is given below-



The *supRuleBase* table has relations with the *exceptionList* table. Any exception figuring in the rule base must have a description in the exception list. The table *supExampleBase* depends on the *exceptionList* and must provide analysis for each

example figuring in the *exceptionList* and marked in the *supRuleBase*. The *dhāturūpa* object depends on the *dhātupāṭha* object while the AVs is a floating object as of now. These linguistic resources are checked for recognition of nouns, and the rules and example bases are searched for analysis. System uses some text/data files whose samples have been given in earlier sections.

5.8 Database Connectivity

The database connectivity is done through Java Database connectivity (JDBC) driver. JDBC Application Programming Interface (API) is the industry standard for database independent connectivity for Java and a wide range of SQL databases. JDBC technology allows using the Java programming language to develop 'Write once, run anywhere' capabilities for applications that require access to large-scale data. JDBC works as bridge between Java program and Database. SQL server 2005 and JDBC support input and output in Unicode, so this system accepts Unicode Devanagari text as well as prints result in Unicode Devanagri too[9].

6 Limitations of the System

6.1 Limitations of the Recognition Process

This system has the following recognition limitations:

- at present, we have approximately verb forms for only 450 commonly found verb roots in the verb database. Though it is very unlikely that ordinary Sanskrit literature will overshoot this list, yet the system is likely to start processing verb forms as nouns if not found in this limited database.
- at this point, the system will wrongly mark prefixed or derived verb-forms as nouns as they will not be found in the verb database. The gains from the *tiṅanta* analyzer will be added here shortly to overcome this limitation.
- currently this work assumes sandhi free text. So, a noun or verb with sandhi is likely to return wrong results. The gains from a separate research on sandhi processing will be used to minimize such errors.
- currently, our AV database has only 519 AVs. It is not enough for AV recognition in ordinary Sanskrit literature. In this case, the system is likely to start processing AVs as nouns, if it is not found in AVs database.
- some forms ending in primary affixes look like nouns while they are AVs. For example: पठितुम्, गत्त्वा, आदाय, विहस्य etc. System will incorrectly recognize and process them as *subantas*.
- many nouns (for example, *śtṛ pratyayānta* in locative singular) look like verbs. These will be wrongly recognized as verbs for example: भवति, गच्छति, पठति, चलति etc. To solve this problem, we will have a hybrid POS category called SUPTIN for those verb forms which are *subantas* as well.

[9] http://java.sun.com/products/servlet/

6.2 Limitations of the Analysis Process

The system has the following analysis limitations:

- same forms are available in the dual of nominative and accusative cases, for example, रामौ, dual of instrumental, dative and ablative cases, for example रामाभ्याम्, plural of dative and ablative cases, for example रामेभ्यः, dual of genitive and locative cases, for example रामयोः. In neuter gender as well, the nominative and accusative singular forms may be identical as in पुस्तकम् (1-1 and 2-1). In such cases, the system will give all possible results as in

रामौ	=	औ	[प्र./ द्वि. द्विव.]
रामाभ्याम्	=	भ्याम्	[तृ./च./पं. द्विव.]
रामेभ्यः	=	भ्यस्	[च./पं. बहुव.]
रामयोः	=	ओस्	[ष./स. द्विव.]
पुस्तकम्	=	सु/अम्	[प्र./द्वि. एकव.]
हरेः	=	ङसि/ङस् [पं./ष. एकव.]	

- some *kṛdanta* forms (generally *lyap, tumun,* and *ktvā* suffix ending) look like nouns (for example - विहस्य पठित्वा, गत्वा, पठितुम्, गन्तुम, नेतुम्, प्रदाय, विहाय etc.). In such cases, the system may give wrong results as:

  विहस्य = विह + ङस् षष्ठी एकवचन
  पठित्वा = पठित्वा + सु प्रथमा एकवचन
  गत्वा = गत्वा + सु प्रथमा एकवचन
  पठितुम् = पठितु + अम् द्वितीया एकवचन
  गन्तुम् = गन्तु + अम् द्वितीया एकवचन

- at this point, system does not have gender information for all PDKs, nor does it attempt to guess the gender. This limitation is going to be minimized by plugging in the *amarakośa* shortly.
- currently this system is giving multiple results in ambiguous cases, because the words as analyzed a single tokens. This will be solved by adding the gains from the research on *kāraka* and gender of nouns which concluded recently.

7 The *TiṅAnta* Analyzer

Verbs constitute an important part of any language. A sentence indispensably requires a verb to convey complete sense. Given the importance of verb and verb phrases in any linguistic data, it is necessary to develop a proper strategy to analyze them. Creating lexical resource for verbs along with other parts of speech is a necessary requirement. Sanskrit is a highly inflectional language. It is relatively free word-order language. The semantic inter-relation among the various components of a sentence is established through the inflectional suffixes.

Scholars have done efforts to analyze Sanskrit verb morphology, both in theory and in computation. Some of the major works are listed below:

- Gérard Huet has developed a lemmatizer that attempts to tag inflected Sanskrit verbs along with other words. This lemmatizer knows about inflected forms of derived stems which are not apparent in the display of the main stem inflection. It, however, does not attempt to lemmatize verbal forms with preverbs but only invert root forms. The site also provides a long list of the conjugated forms of verb-roots in the present, imperfect, imperative, optative, perfect, aorist and future tenses as a PDF document.
- *Prajna* project of ASR Melkote claims to do module generation and analysis of 400 important Sanskrit roots in three voices (Active, Passive and Impersonal), 10 lakāra, 6 tense and 4 moods,
- Aiba (2004) claims to have developed a Verb Analyzer for classical Sanskrit which can parse Sanskrit verb in Present, Aorist, Perfect, Future, Passive and Causative forms. This site actually works only for some verbs and accepts that the results are not reliable,
- *Desika* project of TDIL, Govt. of India claims to be an NLU system for generation and analysis for plain and accented written Sanskrit texts based on grammar rules of Pāṇini's *Aśṭādhyāyī.* It also claims to have a database based on *Amarakośa* and heuristics based on *Nyāya* & *Mīmāmsā Śāstras* and claims to analyze Vedic texts as well,
- RCILTS project at SC&SS, JNU has reportedly stored all verb forms of Sanskrit in a database,
- *Śābdabodha* project of TDIL, govt. of India claims to be an interactive application to analyze the semantic and syntactic structure of Sanskrit sentences,
- The ASR Melcote website reports that a Sanskrit Authoring System is under development at C-DAC Bangalore. The system is supposed to make making tools for morphological, syntactic and semantic analyses with word split programs for sandhi and *samāsa*.
- Cardona (2004) discussed Pāṇini’s derivational system involving aspect of linguistics, grammar and computer science.
- Whitney (2002) listed all the quotable roots of the Sanskrit language together with the tense and the conjugation system.
- Mishra and Jha (2004) describe a module (Sanskrit *Kāraka* Analyzer) for identification and description of *kāraka* according to *Pāṇinian kāraka* formulations.
- Edgren (1885) discussed verb roots of Sanskrit language according to Sanskrit grammarians.
- Joshi (1962) presented linguistic analysis of verb and nouns of Sanskrit language
- Jha and Mishra (2004) proposed *a model for Sanskrit verb inflection identification that would correctly describe verbs in a laukika Sanskrit text.* They presented a module to identify the verb by applying Panini rules in reverse with the help of a relational database.

This module can also be used to identify the types of sentences as active or passive voice with complete reference of the verb.

Present work, which owes a lot to above listed efforts, has some specific features such as:

- The system takes into account the *Pāṇinian* analysis and develops its methodology by applying it into reverse direction.
- It aims at developing a comprehensive strategy so that any *tiṅanta* can be analyzed with the same technique.
- It can be further expanded and modified to recognition and analysis of denominatives
- it is an online servlet-unicode database system with input-output in Unicode only

The front-end of the *tiṅanta* analyzer is as follows -

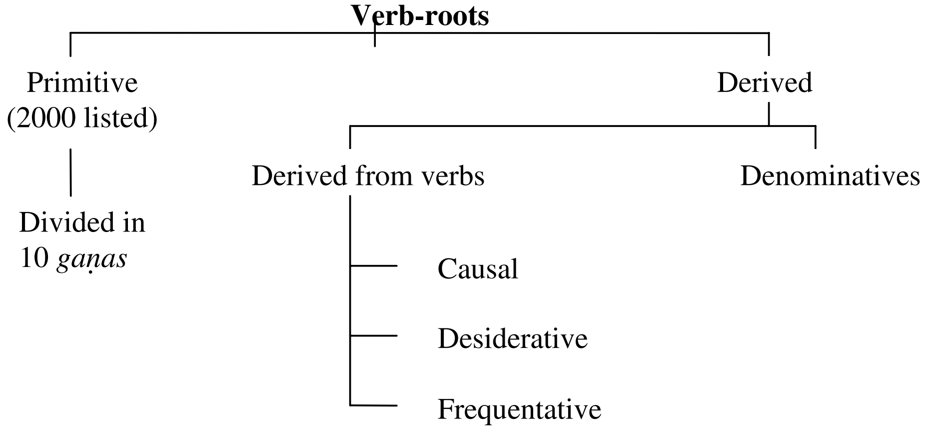


8 Sanskrit Verb-Morphology

Verbs have been of central importance to Sanskrit grammarians. *Yāska* insisted so much on them that he propounded that all the nominal words are derived from verb roots. Verbs convey the sense of becoming[10]. Sanskrit follows a well defined process in the formation of *padas*. Both noun *padas (subanta)* as well as verb *padas (tiṅanta)* have to undergo certain inflectional processes in which various nominal or verbal affixes are added to nominal or verbal base word in order to obtain noun and verbal forms. The process is however more than mere addition as there may occur certain morphophonemic changes in the base as well as in the affix in the process resulting in

[10] *bhāvapradhānamākhyātam (Yāska, Nirukta).*

a usable form. The verb forms are derived from verb-roots or *dhātus*. These *dhātus* are encoded with the core meaning of the verb. These can be primitive[11] or derived[12]. Primitive verb-roots, which are around 2000 in number, have been listed in a lexicon named *dhātupāṭha*. They are divided in 10 groups called *gaṇas*. All the verb-roots of a group undergo somewhat similar inflectional process. Derived verb-roots may be derived from primitive verb-roots or from nominal forms. Prefixes also play an important role as they can change the meaning of a verb root. These roots then have to undergo various inflectional suffixes that represent different paradigms. In this process, the base or root also gets changed. The chart given on the next page gives an overview of Sanskrit verb roots.



8.1 Derived Verb-Roots

8.1.1 Those Derived From Verb-Roots

- **Causatives *(ṇijanta)* -** The causals are formed by adding affix *ṇic* to a primitive verb root. They convey the sense of a person or thing causing another person or thing to perform the action or to undergo the state denoted by the root.
- **Desideratives *(sannanta)* -** Desiderative of a primitive verb root is formed by adding affix *san* to it. It conveys the sense that a person or thing wishes to perform the action or is about to undergo the state indicated by the desiderative form. Any basic verb-root or its causal base may have a desiderative form.
- **Frequentatives *(yaṅanta)* -** Frequentative verbs import repetition or intensity of the action or state expressed by the root from which it is derived. They can be of two types -
    - *Ātmanepada* Frequentative (*yaṅanta*) – affix *yaṅ* is added
    - *Parasmaipada* Frequentative (*yaṅluganta*) – affix *yaṅ* is added but deleted

[11] *bhūvādayo dhātavaḥ* (*Pāṇini* 1/3/1).
[12] *sanādyantā dhātavaḥ* (*Pāṇini* 3/1/32).

An illustration is given below of formation of derived verb-roots from a primitive verb root *bhū*.

bhū (to be)----------- --- (+ *ṇic*) ---- *bhāvay* (to cause someone or something to be)
--- (+ *san*) ---- *bubhū*((to desire to be)
--- (+ *yaṅ*) ---- *bobhūya* (to be repeatedly)
(*yaṅ* deleted) ---- *bobho/bobhav*

These derived verb-roots, however, undergo similar operations, with some specifications, to form verb forms.

8.1.2 Those Derived from Nominal Words

A large number of Sanskrit verb-forms can be derived from nominal words. These are known as *nāmadhātus* (denominatives). Taking a nominal word as head, various derivational suffixes are added to these to form nominal verb-roots. The sense conveyed by the nominal verb root depends upon the suffix added to it. Yet, denominatives commonly import that a person or thing behaves or looks upon or wishes for or resembles a person or thing denoted by the noun. These denominatives, however, can be innumerable as there is no end to nominal words in Sanskrit.

8.2 Process of Formation of Sanskrit Verb Forms

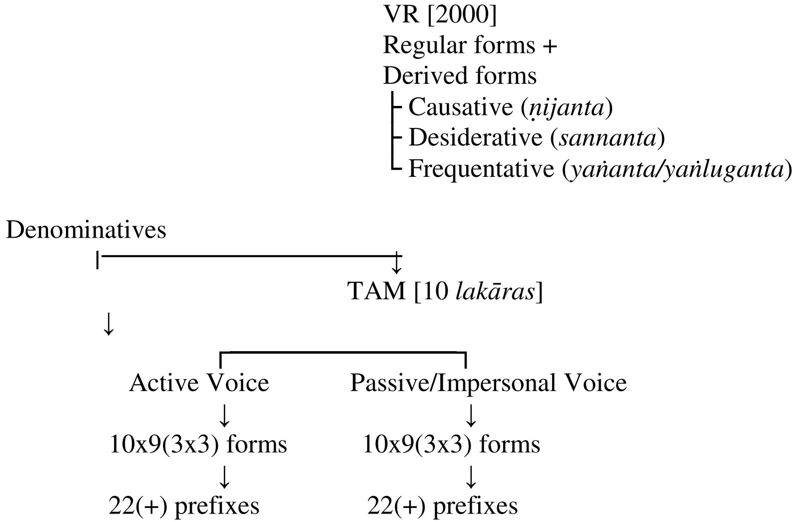
A Sanskrit verb root may take various forms in different inflectional paradigms. Sanskrit has ten *lakāras*, i.e. four moods (Indicative, Imperative, Optative, and Subjunctive) and six tenses (Present, Imperfect, Perfect, Distant Future, Future and Aorist). The *lakāras* are named in C-V-C format. The first consonant *l* signifies that the suffix has to be replaced by *tin* terminations further. The vowels *a, i, o, u, e, o, r* distinguish one *lakāra* from another. Last consonant, either *ṭ* or *ṅ*, signifies different operations. These *lakāras* are added to the root, as primary suffixes, so that it denotes a meaning in the particular tense or mood indicated by that particular *lakāra*.

Verb inflectional terminations or conjugational suffixes are 18 in number. These are divided in two groups – *Parasmaipada* and *Ātmanepada*, each having 9 affixes – a combination of 3 persons x 3 numbers. Thus each of the 18 terminations expresses the voice, person and number. A verb is conjugated in either *pada,* though some of the roots are conjugated in both. For each different *lakāra,* a root is affixed with these 9 terminations in a single *pada*. Again, there are three voices- Active, Passive and Impersonal. Transitive verbs are used in the Active and Passive voices while intransitive verbs are conjugated in the Active and Impersonal voices. The 18 inflection terminations are basically replacement of the *lakāra* or primary suffix. According to Pāṇini, when a *lakāra* is added to a root, it is replaced by 18 terminations. Thereafter, one of the 18 remains to create a verb form.

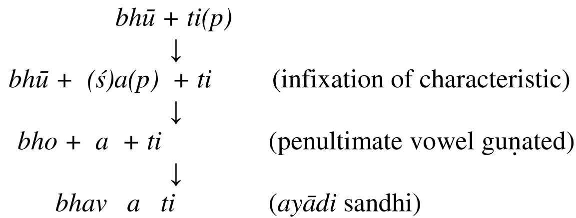
For each separate *lakāra*, the 18 *tin* terminations are replaced by other forms, an illustration of the replacement technique of Pāṇini. Thus *ti, tu, tā, t* etc. are the various replacements of same affix tip in the environment of different *lakāras*.

Then we have certain characteristics *(vikaraṇa)* inserted between the root and the termination. This characteristic can vary according to *lakāra* or the class of the verb root. For four of the *lakāras*, we have *śap* as a characteristic – only for four *gaṇas*.

Addition of one or more of 22 prefixes (*upasargas*) to verb roots can result in more variety of forms. Derivative verb roots, both derived from verb roots as well as nominal words, also follow the same process to form verb forms. There can be some specific rules and exceptions in some cases. The following tree gives a rough estimate of all the possible verb-forms of Sanskrit.[13].



The verb roots of different *gaṇas* adopt certain terminations when *tiṅ* affixes are added to them. Consequently, the verb roots of these classes form verbal bases ending in *'a'*. The *tiṅ* affixation also influences the verb root and it undergoes several morpho-phonemic changes, for example, having *guṇa* operation on the end vowel. The verb root can adopt certain more operations resulting in the final verb-form.



As shown in an example, when suffix *tip* is added to verb-root *bhū*, we obtain *bhavati* as the final verb form. This *bhavati* can be analyzed in *bhav + a + ti*. Here *bhav* is the prepared verbal base whereas *a+ti* is the combination of 'characteristic + conjugational affix.' This can be cited as a common analysis of most Sanskrit verb

[13] Mishra Sudhir K., Jha, Girish N., 2004, *Identifying Verb Inflections in Sanskrit morphology*, In proc. of SIMPLE 04, IIT Kharagpur, pp. 79-81.

forms. The verbal base of a verb root remains same in all its forms whereas the second combination is common for almost all the roots of a single *gaṇa.*

The analysis applies to the first category of derived verb roots as well.

9 Analysis of Sanskrit Verb Forms

9.1 Strategy for Regular Verb Forms

The simplest strategy for regular verb forms can be to store all the possible forms of all the verb roots in any structured form. But given the enormity of Sanskrit verb-roots and the multiplicity of inflectional paradigms, this approach is far from being practical. A better approach may be arrived at by following the analytical method.

As illustrated above, Sanskrit verb forms are a blend of multiple morphemes which contain relevant information. Analytically, we can say that the first element is the conjugational affix that remains at the end of every verb form. These affixes have encoded information of *pada* (though it is determined by the root), *lakāra*, person and number. Thus terminations can serve as the most important thing to let us know about the paradigm information of any verb form. They can be a tool to identify a verb form in a given text. The terminations, as they are basically replacements of 18 original *tiṅ* affixes in different *lakāras*, differ among themselves according to the *lakāra*. However, in each *lakāra* they are similar for all the verb roots of various groups, leaving some exceptions. So, *ti* can be used to identify any verb form of present tense in *parasmaipada*. But some terminations can vary among themselves for a group of *gaṇas*. Then again, the terminations may be changed due to morphophonemic environment, *tā* affix of *luṭ lakāra* changing to *ṭā* with roots like *yaj*.

Further left we have the remaining morphemes of the various characteristics and increments inserted between the verb root and terminations, in the process of their formation explained above. So*, bhvādigaṇa* verb forms, in conjugational *lakāras*, have '*a'*- a result of *śap* characteristic; *svādi* roots have *no*, *nu* or *nv* - all of them remaining morphemes of *śnu*. Some roots like that of *adādi* have no such characteristic sign infixed in them.

Then we have the modified stem of the verb root at the right end of the verb form. The modification can be that of *guṇa*, *vṛddhi* or any other. Generally a root adopts a common stem in all the forms for both the *padas* in conjugational *lakāras*. So, *bhav* is the stem for all the *parasmai* forms in the conjugational *lakāras*. But there are exceptions to it to the extent that four or five types can be found among nine forms of a single *lakāra-pada*.

Here, the first morpheme- the *tiṅ* termination is common among all the verb forms of a particular *pada*-*lakāra*-person-number combination. Second constituent- the characteristic (existing in the form of its remaining morpheme) and increments inserted in between may differ, yet being almost the same in a particular group. The third constituent- the modified verb-root is particular in the strict sense. In the

analysis, the recognition of the *tiṅ* termination will identify a word as a verb form and find out its *pada-lakāra*-person and number. The second morpheme can, in many cases, be helpful to recognize the *gaṇa* of a particular root because the characteristics in a *lakāra* are determined by the *gaṇa* that the root belongs to.

Thus the core of the analytical approach is that each *tiṅanta* verb form can be analyzed to form a unique combination of verbal stem + *tiṅ* termination; and we store both of these constituent parts in separate tables. So, when it is to be analyzed, its constituent morphemes are recognized and identified with the help of pre-stored structured data.

An example of this strategy is shown in the table given below. The first column demonstrates the representative verb root of each class. When the verbal affix *tip* is added to each of them, it undergoes certain morphological operations and results in the usable *tiṅanta* verb form listed in the second column. This is the forward Pāṇinian process. The next column demonstrates the reverse Pāṇinian approach for analysis of verb forms. In the second column every form has *ti* ending. When we remove this ending along with the conjugational affix, we obtain the storable verbal base. Every verb form can be analyzed similarly in ending and remaining verbal base.

Verb-root		**Verb-form**		**Verb-base**
bhū		*bhavati*		*bhav (-ati)*
ad		*atti*		*at(-ti)*
hu		*juhoti*		*juho(-ti)*
div		*dīvyati*		*dīvy(-ati)*
su	+ *ti* →	*sunoti*	- *ti* →	*sun(-oti)*
tud		*tudati*		*tud(-ati)*
chid[14]		*chinatti*		*chinat(-ti)*
tan		*tanoti*		*tan(-oti)*
krī		*krīṇāti*		*krī(-ṇāti)*
cur		*corayati*		*coray(-ati)*

9.2 Database Tables for Verb Analysis

The database tables given below demonstrate the structure of storage of all possible verbal bases of a verb root. As a sample data, five verb roots of different *gaṇas* have been taken -

Table of Verbal Bases

root	gaṇa	pada	seṭ/ aniṭ/ veṭ	lakāra	Verbal Bases				
					Regular	Causal	Desider.	Frequentative	
								Ātmane	Parasmai
bhū	*bhvādi*	*parasmai*	*seṭ*	*laṭ/loṭ/vli*	*bhav,*	*bhāvay*	*bubhūṣ*	*bobhūy*	*bobhav ,bobho*
				liṭ	*babhūva*	*bhāvayāñ/m*			

[14] *rudh* shows an exceptional behaviour, so *chid* has been taken.

				laṅ	*abhav*	*abhāvay*			
				ali	*bhū*	*bhāv*			
				luṅ	*abhū*	*abībhav*			
ad	*adādi*	*paras mai*	*seṭ*	*laṭ/loṭ/ vli*	*ad,at*	*āday*	*jighats*	-	
				liṭ	*jaghāsa, jaghasa jakṣa,āda*	*ādayñ*			
				laṅ	*ād,āt*	*āday*			
				ali	*ad*	*ād*			
				luṅ	*aghas*	*ād*			
hu	*juhotyādi*	*paras mai*	*aniṭ*	*laṭ/loṭ/ vli*	*juho, juhu,juhv*	*hāvay*	*juhūṣ*	*johūy*	*joho*
				liṭ	*juhavāñ/m, juhāva, juhuv*				
				laṅ	*ajuho,ajuhu*				
				ali	*hū*				
				luṅ	*ahau*				
div	*divādi*	*paras mai*	*seṭ*	*laṭ/loṭ/ vli*	*dīvy*	*devay*	*dideviṣ*	*dedīvya*	-
				liṭ	*didev,didiv*				
				laṅ	*adīvy*				
				ali	*dīv*				
				luṅ	*adev*				

The second table illustrates the structure of the storage of verbal terminations of five *gaṇas* in both *padas* for *laṭ lakāra*. More than one termination in a single box has been separated.

Table of Verbal Affixes

lakāra	pada/gaṇa		I per.			II per.			III per.		
			Sing	Dual	Plu.	Sing.	Dual	Plu.	Sing	Dual	Plu.
laṭ	**parasmai**		***ti/ ati/ oti/ ūti***	***taḥ/ ataḥ/ utaḥ***	***nti/ anti/ vanti***	***si/ṣi/ asi/ aṣi/oṣi/ osi***	***thaḥ/ athaḥ/ uthaḥ***	***tha/ atha/ utha***	***mi/ āmi/ omi***	***vaḥ/ āvaḥ/ uvaḥ***	***āmaḥ/ maḥ/ umaḥ***
	ātmane		***te/īte /ūte ute/ īte***	***ete aate/ īyāte/ uvāte***	***ante/ ate/ īyate/ uvate***	***se/ īṣe/ uṣe ase***	***ethe/ āthe/ īyāthe/ uvāthe***	***adhve/ īdhve/ udhve***	***e/ īye/ uve***	***āvahe/ī vahe/ uvahe***	***āmahe/ īmahe/ umahe***

laṅ	**parasmai**		***t/at***	***tām/ atām***	***n/an***	***ḥ/aḥ***	***tam/ atam***	***ta/ ata***	***m/am***	***āva***	*āma*
	ātmane		***ata***	***etām/***	***anta***	***athāḥ***	***ethām***	***adhvam***	***e***	***āvahi***	*āmahi*
luṅ	**parasmai**		***īt***	***iṣṭām***	***iṣuḥ***	***īḥ***	***iṣṭam***	***iṣṭa***	***iṣam***	***iṣva***	*iṣma*
	ātmane		*iṣṭa*	*iṣātām*	*iṣata*	*iṣṭhāḥ*	*iṣāthām*	*idhvam/ idhvam*	*iṣi*	*iṣvahi*	*iṣmahi*

For identification and analysis, the suffixes should be given a descending character sequence. So *ti* of *iṣyati* in *bhaviṣyati* cannot create any ambiguity.

10 Problems and Possible Solutions

- Verb forms which have no mark of termination left in the end are difficult to identify with the proposed module. So *bhava*, *babhūva* and other alike forms are to be stored separately.
- Some forms which are not *tiṅanta* but are similar to them like *bhavati, bhavataḥ* which are singular and dual of *bhū* in present *parasmai* third person, and also locative singular and ablative/relative singular of nominal root *bhavat.* The resolution of ambiguity here will demand involvement of semantic and syntactic analysis.
- Denominatives are formed by deriving verbal base from nominal base with the help of affixes such as *kyac, kāmyac, kyaṣ, yak, kyaṅ* etc. and then adding various verbal terminations to these verbal bases. Thus they undergo same operations and processes as regular and derived verb forms. Still there analysis is difficult due to two reasons. Firstly, nominal bases can be innumerable and thus the above stated strategy of storing the bases of all the nominative verbal bases is impossible in this case. One has to follow the rule based analytical approach. The verbal terminations can be determined with the help of affix tables as denominatives are affixed with same verbal affixes. The remaining base, however, has to be analyzed in order to infer the nominal base of that denominative. As there are some common rules to derive the verbal stem from nominal base, we can develop an analysis rule based module to identify the nominal root and can find its meaning with the help of a lexicon.

- Addition of prefixes to the verbal bases may cause morphological as well as semantic change to a verbal form. To identify one or more prefix in a verbal form, all the prefixes have to be stored in a database table along with their meaning. The system will have to check the input verbal form from left to identify single or combined prefixes. A prefix can happen to completely modify the meaning of a verb. So, creating a separate table that stores the altered meanings of various roots, when affixed with certain prefix, may be helpful in this case.

11 Conclusion

The proposed strategy to analyze Sanskrit verb forms in given text is different from existing works in many ways. It works with a reverse Pāṇinian approach to analyze *tiṅanta* verb forms into there verbal base and verbal affixes. The methodology accepted to create database tables to store various morphological components of Sanskrit verb forms is clearly in line with the well defined and structured process of Sanskrit morphology described by Pāṇini in his *Aṣṭādhyāyī*. It comprehensively includes the analysis of derived verb roots also. Even in the case of verb roots derived from nominal words, the table of affixes can provide assistance in order to separate the denominative verbal base from the verbal terminations.

References

1. Bharati, A., Chaitanya, V., Sangal, R.: A Computational Grammar for Indian languages processing. Indian Linguistics Journal, 52, 91–103 (1991)
2. Bharati, A., Chaitanya, V., Sangal, R.: Natural Language Processing: A Paninian Perspective. Prentice-Hall of India, New Delhi (1995)
3. Cardona, G.: Pāṇini's syntactic categories. Journal of the Oriental Institute, Baroda (JOIB) 16, 201–215 (1967)
4. Cardona, G.: Panini: his work and its tradition, vol. 1. Motilal Banarasidas, Delhi (1988)
5. Cardona, G.: Some Questions on Pāṇini's Derivational system. In: Procs of SPLASH, iSTRANS, p. 3. Tata McGraw-Hill, New Delhi (2004)
6. Daniel, J., Martin, J.H.: Speech and Languages Processing. Prentice-Hall, New Delhi (2000)
7. Edgren, A.H.: On the verbal roots of the Sanskrit language and of the Sanskrit grammarians. Journal of the Americal oriental Society 11, 1–55 (1885)
8. Huet, G.: Towards Computational Processing of Sanskrit, Recent Advances in Natural Language Processing. In: Proceedings of the International Conference ICON, Mysore, India (2003)
9. Jha, G.N., et al.: Towards a Computational analysis system for Sanskrit. In: Proc. of first National symposium on Modeling and Shallow parsing of Indian Languages at Indian Institute of Technology Bombay, pp. 25–34 (2006)
10. Jha, G.N.: A Prolog Analyzer/Generator for Sanskrit Noun phrase Padas. Language in India 3 (2003)
11. Jha, G.N.: Generating nominal inflectional morphology in Sanskrit. In: SIMPLE 2004, IIT-Kharagpur Lecture Compendium, Shyama Printing Works, Kharagpur, WB, pp. 20–23 (2004)

12. Jha, G.N.: Morphology of Sanskrit Case Affixes: A computational analysis, M.Phil dissertation submitted to J.N.U., New Delhi (1993)
13. Jha, G.N.: The system of Panini. Language in India 4(2) (2004)
14. Joshi, S.D.: Verbs and nouns in Sanskrit. Indian linguistics 32, 60–63 (1962)
15. Kapoor, K.: Semantic Structures and the Verb: a propositional analysis. Intellectual Publications, New Delhi (1985)
16. Mishra, S.K., Jha, G.N.: Identifying Verb Inflections in Sanskrit morphology. In: Proc. of SIMPLE 2004, IIT Kharagpur, pp. 79–81 (2004)
17. Mishra, S.K., Jha, G.N.: Sanskrit Karaka Analyzer for Machine Translation. In: SPLASH proc. of iSTRANS, pp. 224–225. Tata McGraw-Hill, New Delhi (2004)
18. Ruslan, M.: The Oxford Handbook of Computational Linguistics. Oxford University Press, Oxford
19. Mishra, N. (ed.): Kashika of Pt.Vamana and Jayaditya. Chaukhamba Sanskrit sansthan, Varanasi (1996)
20. van Nooten, B.A.: Pāṇini's replacement technique and the active finite verb. University of California, Berkeley
21. Sharma, R.N.: The Aṣṭādhyāyī of Pāṇini. Munshiram Manoharlal Publishers Pvt. Ltd., Delhi (2003)
22. Shastri, B.: Laghusiddhantakaumudi (1st part), Bhaimee Prakashan, 537, Lajapatrai Market, New Delhi
23. Shastri, S.D.: The Mādhavīya Dhātuvṛtti by Sāyaṇacārya. Tara Book Agency, Varanasi (2000)
24. Subash, Jha, G.N.: Morphological analysis of nominal inflections in Sanskrit. In: Platinum Jubilee International Conference, L.S.I. at Hyderabad University, Hyderabad, p. 34 (2005)
25. Subash: Machine recognition and morphological analysis of Subanta-padas, M.Phil dissertation submitted to J.N.U., New Delhi (2006)
26. Upadhye, P.V.: Dhāturūpacandrikā. Gopal Narayen & Co., Bombay (1927)
27. Whitney, W.D.: History of Sanskrit Grammar. Sanjay Prakashan, Delhi (2002)

Web References

- IIIT, Hyderabad, `http://www.iiit.net/ltrc/Publications/Techreports/tr010/anu00kbcs.txt` (accessed: 22nd April 2007).
- Peter M. Scharf and Malcolm D. Hyman, `http://sanskritlibrary.org/morph/` (accessed: 12 August 2006).
- Huet's site `http://sanskrit.inria.fr/`
- Prajna system, ASR Melcote, `http://www.sanskritacademy.org/Achievements.htm`
- Aiba, Verb Analyzer for classical Sanskrit, `http://wwwasia.human.is.tohoku.ac.jp/demo/vasia/html/`
- Desika, TDIL, Govt. of India, `http://tdil.mit.gov.in/download/Desika.htm`
- RCILTS, JNU, `http://rcilts.jnu.ac.in`
- Shabdabodha, ASR, Melcote, `http://vedavid.org/ASR/#anchor2`

Semantic Processing in Pāṇini's Kāraka System

Girish Nath Jha[1] and Sudhir K. Mishra[2]

[1] Special Center for Sanskrit Studies, J.N.U. New Delhi-110067
girishj@mail.jnu.ac.in
[2] CDAC Pune (Applied AI Group), Pune-411005
sudhirkumarmishra@gmail.com

Abstract. Pāṇini's grammar is widely known for its formal treatment of the Sanskrit language. Many scholars (Jha 2004) have earlier taken a systemic view of Pāṇini and have argued that Pāṇini's system is easily implementable. However, on a closer look, several complications arise, especially in Pāṇini's recourse to semantics in many of the vidhi and saṁjñā rules. This seems to happen more in the kāraka prakaraṇa than in other components. The authors of this paper have highlighted the challenges in implementing some of the semantic aspects of Pāṇini's kāraka system. For example, the semantic conditions of being most desired by the agent (*karturīpsitatamam*) and of being most effective (*sādhakatamam*) are difficult to formalize in the rules specifying the semantic conditions for an object's being termed *karman* and *karaṇa* respectively. Likewise, the semantic conditions of being that which the agent approaches with the direct object (*karmaṇā yamabhipraiti*), and being the one pleased in relation to verbs of pleasing (*rucyarthānām prīyamāṇaḥ*) are difficult to formalize in rules specifying semantic conditions for an object's being termed *sampradāna.* The paper also looks at possible strategies to handle such situations, and presents pseudo-code like translations of some kāraka rules.

Keywords: vārtika, kāraka, vibhakti , upapada vibhakti, karmapravacanīya, semantics, śiva-sūtra, *Ṛgveda, Nirukta,* Brāhmaṇa, śikṣā, prātiśākhya, kartṛ, karma, karaṇa, sampradāna, apādāna, adhikaraṇa, prathamā, dvitīyā, tṛtīyā, caturthī, pañcamī, ṣaṣṭhī, saptamī, ākāṅkṣā, yogyatā, vivakṣā.

1 Indian Linguistic Tradition: Emphasis on Semantics

Semantics and phonetics have been the basis of all linguistic analysis in the Sanskritic tradition before, during, and after Pāṇini. According to Nayar (2002: 99), 'it was always held that all linguistic analysis is a step-by-step progress from sound to sense involving a series of contextualizations'. The Vyākaraṇa Vedāṅga which culminated in Pāṇini, provided steps for semantic interpretation through carefully crafted morpho-syntactic procedures often bordering on semantic conditions. Compared to earlier generative linguistics where semantics was given a lesser or no role, the Indian tradition has emphasized a logical movement from sound to meaning. An utterance is contextualized, and the context is considered to be a playfield of semantics. We will do a

G. Huet, A. Kulkarni, and P. Scharf (Eds.): Sanskrit CL 2007/2008, LNAI 5402, pp. 239–252, 2009.

quick survey of some relevant literature since the *Ṛgveda* to ascertain the role of semantic studies in the Indian linguistic tradition.

1.1 The Ṛgveda

The *Ṛgveda* is probably the earliest documentation on language. It emphasizes the correctness of speech and refers to it in the following terms:

- source of happiness
- means of attaining desired ends
- revealer of true knowledge
- permeates every aspect of life
- nothing exists beyond language
- good speech is original and creative

While many similar references are found at many places in the *Ṛgveda*, the most appropriate references to language are in the famous *Vāk Sūkta* (X.125). Let us look at some of its references to language below:

- language is everywhere
- it is a cognitive instrument
- a source of power
- a social force
- the very basis of existence

The *Vāk Sūkta* seems to be emphasizing the social function of language the very basis of which is the correct interpretation.

1.2 Yāska

Yaska's role in textual interpretation of Vedic texts through his etymological treatise the *Nirukta* highlights the fact that there was significant emphasis on getting the right meaning of the texts.

1.3 The Brāhmaṇa Texts

Among the various Brāhmaṇa texts, the *Kauśītakī Brāhmaṇa* praises the Kuru-Pāñcāla speech as the abode of vāk. While clarifying the relation between speech and mind, the text says that speech is impossible without mind, that speech is preceded by mind, and expresses mind (the *Pañcaviṁśa Brāhmaṇa*). According to the *Jaiminīya Brāhmaṇa*, speech is the canal of mind. The *Śatapatha Brāhmaṇa* states that the mind is limited, and the speech is less comprehensive than the mind.

1.4 The Upaniṣads

The Upaniṣadas emphasize the importance of correct speech and its social value. Among them, the *Praśna, Māṇḍūkya, Bṛhadāraṇyaka, Kena* and *Chāndogya* Upaniṣadas elaborate on the physics and metaphysics of speech in this context.

1.5 The Śikṣā- Prātiśākhya Texts

The *Vājasaneyī Prātiśākhya* (also called the *Śukla Yajurveda Prātiśākhya*) emphasizes the importance of proper recitation (*samyak pāṭha*). The *Pāṅini-Śikṣā* emphasizes the śabdārtha as the goal of phonetics.

प्रसिद्धमपि शब्दार्थमविज्ञातमबुद्धिभिः
पुनर्व्यक्तीकरिष्यामि वाच उच्चारणे विधिम्
prasiddham api śabdārtham avijñātam abuddhibhiḥ
punarvyaktī kariṣyāmi vāca uccāraṇe vidhim (PS.2)

1.6 Patañjali

Patañjali states that one of the goals of linguistic study in Indian tradition was defense of scriptures (rakṣā). This goal could be achieved through grammar by explaining the meaning of scriptures to those who thought they were meaningless. He also emphasizes the right knowledge (samyak jñāna) and precise usage (suṣṭhu prayoga) of a word.

1.7 Semantic Roadblocks in Pāṇini

A complete computer simulation of Pāṇini has semantic hurdles to cross. This is due to the fact that the very basis of Pāṇini's processing is semantics. Cardona (2004: 3) has also pointed to semantics being the basis of all derivations in Pāṇini. Ramanujam (2002: 72) while describing his Desika application has expressed his frustration at semantic issues in Pāṇini, stating, 'there are quite a few issues which have no satisfactory method of handling at the moment. These obviously pertain to the semantics covered/sought to be conveyed by different word categories, as described by sage Pāṇini.' Pāṇini's morpho-syntactic rules as seen in inflectional morphology (subanta, tiṇanta), derivational morphology (kṛdanta, taddhita, strī pratyaya, samāsa) and syntactico-semantic rules (kāraka) heavily depend on semantic conditions thereby making their implementation difficult.

2 Pāṇini's kāraka-vibhakti System

Etymologically *kāraka* is the name given to the relatum of the action signified by the verb in a sentence. It means 'that which brings about' or 'doer'. Pāṇini (*kārake* 1.4.23) introduces kārakas probably as adhikāra. Normally technical terms are introduced in the nominative case but in 1.4.23, the term *kārake* is introduced in the locative case. Pāṇini discusses the entire gamut of kāraka-vibhakti relations in three sections of the *Aṣṭādhyāyī:*

- kāraka sūtra (1.4.23 – 1.4.55)
    - 33 kāraka-sūtras in the following sequence: apādāna, saṃpradāna, karaṇa, adhikaraṇa, karman and kartṛ
- vibhakti sūtra (2.3.1 - 2.3.73)
- karma-pravacanīya (1.4.83 – 1.4.98)

Patañjali defines the term *kāraka* in 1.4.23 as *karoti iti kārakam* (that which brings out or accomplishes action). Kātyāyana takes *kārake* as a saṁjñā sūtra. Bhartṛhari (Bhatta 1992) argues that kāraka is the capacity to produce an action. Nāgeśa interprets kāraka to be something having the capacity to accomplish an action. Jagadiśa (Bhatta 1992) in the *Śabdaśaktiprakāśikā* states that the meaning expressed by the particular post-nominal affix becomes the kāraka, if such a meaning is related to the action expressed by the verb. Giridhara (*Vibhaktyarthanirṇaya*) adds some constraints to the theory of kāraka outlined by Jagadiśa. According to him the meaning of the post-nominal affixes can be regarded as kāraka provided that

- they are invariably related to an action
- they are never related to the meaning of any nominal stem
- they are not referred to by the post-nominal affixes that are added to a noun on account of the presence of another word

The *Siddhānta Kaumudī*'s section on cases is called *Vibhaktyartha Prakaraṇa* (the section dealing with the semantics of vibhaktis) by many scholars. Cardona (2004: 3) calls kārakas 'participants in actions'. According to Ghildiyal (1962), kārakas are the meanings of vibhaktis. Kārakas are the meanings, and vibhaktis are their markers. There are three kinds of vibhaktis: kāraka vibhaktis, and special cases (including upapada vibhaktis).

- kāraka vibhaktis: kartā, karma, etc.
    - Special cases: In the context of some words, dvitīyā vibhakti is ordained. For example, *kāla* and *adhva* (*kālādhvanor atyantasaṁyoge* P.2.3.5). The upapada vibhaktis can also be included in this category. For example, the rule *abhitaḥparitaḥsamayānikaṣāhāpratiyoge'pi* allows dvitīyā vibhakti in the case of an upapada used with *abhitaḥ*, *paritaḥ*, *samayā*, *nikaṣā*, *hā* or *prati*.

3 Formalizing Pāṇini's kāraka Rules

The following sections attempt an algorithmic representation of the kāraka sūtras with many semantic conditions difficult to formalize. The formulation of rules related to the kartṛ kāraka is as follows:

3.1 kartṛ

Rule	1.4.54 : स्वतन्त्रः कर्ता
Condition	kāraka independent of others
Result	kartṛ kāraka
Code	{ कर्ता ([स्वतन्त्र]) }

Requirement – a formal definition of 'independence'

kartṛ as karman

Rule	1.4.52: गतिबुद्धिप्रत्यवसानार्थ शब्दकर्माकर्मकाणामणिकर्ता स णौ
Condition	causatives of the verbs in the sense of *gati, buddhi, prtyāvasāna*

Result	kartṛ becomes karman
Code	{ कर्म (कर्ता [णिजन्त(शब्दकर्मक, बुद्धि, प्रत्यवसानार्थक धातु, अकर्मक धातु)]) }
Example	शत्रून् अगमयत् स्वर्गम्

Requirement – an explicit listing of verbs meaning *gati, buddhi, pratyavasāna*

3.2 karman

Rule	1.4.49 : कर्तुरीप्सिततमं कर्म
Condition	most sought by kartṛ
Result	karman kāraka
Code	{ कर्म (ईप्सिततम् [कर्ता]) }
Example	माषेषु अश्वं बध्नाति

Rule	1.4.50 : तथायुक्तं चानीप्सितम्
Condition	something not sought by kartṛ
Result	Karman
Code	{ कर्म (अनीप्सितम् [कर्ता]) }
Example	ग्रामं गच्छन् तृणं स्पृशति

Requirement – definitions of 'desired', 'most desired' and 'not desired' in the context of agent

3.3 karaṇa

Rule	1.4.42 : साधकतमं करणम्
Condition	The factor most effective in the accomplishment of the action
Result	karaṇa kāraka
Code	[करण (साधकतम)]
Example	कलमेन लिखति

Requirement – defining the instrument in the accomplishment of an action

3.4 saṁpradāna

Rule	1.4.32 : कर्मणा यमभिप्रैति स सम्प्रदानम्
Condition	That which the agent wants to approach with the object of the action of giving
Result	sampradāna kāraka
Code	{ सम्प्रदान (यमभिप्रैति [कर्मेण]) }
Example	राजा विप्राय गां ददाति

Requirement – formal definition of अभिप्रैति (of the agent through his/her action)

Rule	1.4.33 : रुच्यर्थानां प्रीयमाणः
Condition	The one being pleased in relation to verb roots having the sense of *ruc*

Result	sampradāna kāraka
Code	{ सम्प्रदान (प्रीयमाण [अभिलाषार्थक धातु]) }
Example	हरये रोचते भक्तिः

Requirement – defining प्रीयमाण in the sense of रुच्

Rule	1.4.34 : श्लाघह्नुङ्स्थाशपां ज्ञीप्स्यमानः
Condition	The person who is desired to be cognizant of the praise when verbal roots *ślāgh* (to praise), *hnuṅ* (to hide), *sthā* (to stay) and *śap* (to swear) are used.
Result	sampradāna kāraka
Code	{ सम्प्रदान (ज्ञीप्स्यमान [श्लाघ्, ह्नुङ्, स्था, शप्]) }
Example	गोपिस्मरात् कृष्णाय श्लाघते

Requirement – defining the ज्ञीप्स्यमान (desired receiver of praise etc.)

Rule	1.4.35 : धारेरुत्तमर्णः
Condition	In the case of the use of the causal of *dhṛñ* (owe), the creditor
Result	sampradāna kāraka
Code	{ सम्प्रदान (उत्तमर्ण [णिजन्त धृञ् धातु]) }

Requirement – defining the उत्तमर्ण (the creditor)

Rule	1.4.37 : क्रुधद्रुहेर्ष्यासूयार्थानां यं प्रति कोपः
Condition	To whom anger is directed when verb roots having the signification of *krudh* (to be angry), *druh* (to wish harm to), *īrṣya* (not to tolerate) and *asūya* (to invent fault) are used
Result	sampradāna kāraka
Code	{ सम्प्रदान (यं प्रति कोपः [क्रुध्, द्रुह्, ईर्ष्य, असूयार्थक धातु]) }
Example	हरये क्रुद्ध्यति द्रुह्यति ईर्ष्यति असूयति वा

Requirement – defining the यं प्रति कोपः (to whom anger is directed)

Rule No.	1.4.39
Rule	राधीक्ष्योर्यस्य विप्रश्नः
Condition	To whom many enquiries are made when verbal roots *rādh* (to prophesy) or *īkṣ* (observe) are used.
Result	sampradāna kāraka
Code	{ सम्प्रदान (विप्रश्नः [राध्, ईक्ष्]) }
Example	कृष्णाय राध्यति ईक्षते वा

Requirement – defining the विप्रश्नः (to whom many enquiries are made)

3.5 apādāna

Rule	1.4.24 : ध्रुवमपायेऽपादानम्
Condition	The fixed point in relation to moving away
Result	apādāna kāraka
Code	{ अपादान (ध्रुव [अपाय]) }
Example	वृक्षात् पत्राणि पतन्ति

Requirement – defining the ध्रुव (fixed point) and अपाय (separation from a fixed point)

Rule	1.4.25 : भीत्रार्थानां भयहेतुः
Condition	in the sense of *bhī* (fearing) or *trā* (protecting), the item which causes fear
Result	apādāna kāraka
Code	{ अपादान (कारण [भय, त्राणार्थक धातु]) }
Example	चोरात् बिभेति त्रायते वा

Requirement – defining the भयहेतुः (cause of fear)

Rule	1.4.26 : पराजेरसोढः
Condition	The item which is unbearable when verb base *ji* (to win) is used with the preverb *parā*.
Result	apādāna kāraka
Code	{ अपादान (असोढ [परा + जि धातु]) }
Example	अध्ययनात् पराजयते

Requirement – defining the असोढ (unbearable)

Rule	1.4.28 : अन्तर्धौ येनाद येनादर्शनमिच्छति
Condition	The person by whom one wishes not to be seen through the use of something which comes in between
Result	apādāna kāraka
Code	{ अपादान (अदर्शनम् [अन्तर्धान]) }
Example	मातुर्निलीयते कृष्णः

Requirement – defining the अदर्शनम् (hiding)

3.6 adhikaraṇa

Rule No.	1.4.45
Rule	आधारोऽधिकरणम्
Condition	if there is a locus of the action
Result	adhikaraṇa kāraka
Code	(अधिकरण [आधार])

Example ग्रामे वसति

Requirement – defining the आधार (locus)

Generally saptamī vibhakti (2.3.36) is used in adhikaraṇa kāraka, but tṛtīyā (2.3.44), pañcamī (2.3.42), and ṣaṣthī (2.3.38) vibhaktis are also used in place of or along with saptamī under other conditions.

4 Formalizing Pāṇini's vibhakti Rules

The following sections attempt a formal representation of some of the vibhakti rules of Pāṇini and try to underline the semantic problems therein.

4.1 prathamā

Rule No.	2.3.46
Rule	प्रातिपदिकार्थलिङ्गपरिमाणवचनमात्रे प्रथमा
Condition	if prātipadikārtha (nominal stem meaning), liṅga (gender), parimāṇa (measure) or vacana (number) only is to be expressed.
Result	prathamā vibhakti
Code	{ प्रथमा (प्रातिपदिकार्थ [जाति, व्यक्ति], लिङ्ग, परिमाण, वचन) }

Requirement – a lexicon of words meaning *prātipadika, liṅga, parimāṇa*, and *vacana*

4.2 dvitīyā

Rule No.	2.3.5
Rule	कालाध्वनोरत्यन्तसंयोगे
Condition	If atyantasaṁyoga (continuous connection) is signified in the context of words denoting kāla (time) and mārga (path).
Result	dvitīyā vibhakti
Code	{ द्वितीया (कालवाचक, मार्गवाचक शब्द [संयोग]) }

Requirement – defining *atyantasaṁyoga* (continuous connection) and listing words denoting *kāla* (time) and *mārga* (path).

Rule No.	2.3.8
Rule	कर्मप्रवचनीययुक्ते द्वितीया
Condition	if a word is used in construction with a karmapravacanīya
Result	dvitīyā vibhakti
Code	{ द्वितीया (कर्मप्रवचनीय) }

Requirement – defining *karmapravacanīya*

5 karmapravacanīyas

The following sections present a formalization of some of the karmapravacanīya related *sūtras*

5.1 anu

Pāṇini applies saṁjñā *karmapravacanīya* to *anu* under four semantic conditions and allows for dvitīyā vibhakti [2.3.8].

Rule No.	1.4.84
Rule	अनुर्लक्षणे
Condition	when *anu* cosignifies a lakṣaṇa (characteristic mark).
Result	karmapravacanīya

Rule No.	1.4.85
Rule	तृतीयार्थे
Condition	when *anu* cosignifies the sense of the tṛtīyā (third triplet of *sup*).
Result	karmapravacanīya

Rule No.	1.4.86
Rule	हीने
Condition	when *anu* cosignifies hīna (less, inferior).
Result	karmapravacanīya
Rule No.	1.4.90
Rule	लक्षणेत्थम्भूताख्यानभागवीप्सासु प्रतिपर्यनवः
Condition	when *anu* cosignifies a lakṣaṇa (characteristic mark), itthambhūtākhyāna (characterizing a as b), a bhāga (share) or vīpsā
Result	karmapravacanīya

Combined pseudo-code of the sūtras 1.4.84-86 and 1.4.90 can be given as follows:

```
{
        कर्मप्रवचनीय
                ( अनु
                        [
                                लक्षणार्थक
                                सहभाव, साहित्यार्थक
                                हीनार्थक
                                लक्षणार्थक, इत्थम्भूताख्यान, भाग, वीप्सार्थक
                        ]
                )
        द्वितीया विभक्ति
}
```

Requirement – defining *lakṣaṇa*, *hīna*, *itthambhūtākhyāna*, *bhāga*, *vīpsā*.

The subsequent karmapravacanīyas (1.4.87-98) require a formal account of terms like वर्जन, अनर्थक, प्रतिनिधि, मर्यादा , प्रतिदान, समुच्चय etc and their senses. They either require defining concepts or compiling huge sense-lexicons.

6 kāraka-vibhakti Mapping

While the details of kāraka-vibhakti relations are discussed in three sections of the *Aṣṭādhyāyī* as mentioned in section 2 above, Table 1 summarizes the major relations discussed in this paper.

Table 1. Kāraka-vibhakti relations in the *Aṣṭādhyāyī*

kāraka	sūtra	Explanation	vibhakti/s
kartṛ	P 1.4.54	One which is **independent** (the most important source) in any action and executes the action	P2.3.46 → prathamā
			P 2.3.18 → tṛtīyā
			P 2.3.65 → ṣaṣthī
karman	P 1.4.49	which is **most desired** of agent through his action	P 2.3.2 → dvitīyā
			P 2.3.12 → caturthī
			P 2.3.65 → ṣaṣthī
karaṇa	P 1.4.42	that which is **most instrumental** in accomplishment of an action	P 2.3.18 → tṛtīyā
sampradāna	P 1.4.32	**whosoever the agent approaches** for the object of the act of giving/benefit	P 2.3.13 → caturthī
apādāna	P 1.4.24	that from which/where **separation/departure is meant**	P 2.3.28 → pañcamī
adhikaraṇa	P 1.4.45	that which is the **locus/ substratum** of the action	P 2.3.36 → saptamī

7 Problems in Implementation

The implementation issues with respect to kārakas can be classified in three categories as follows:

- vivakṣā dependent operations
- vārtikas limiting/extending rules
- Semantic conditions necessary for implementing a rule

Some of these may be difficult to implement with the current state of computing algorithms and technologies. Let us see each of them.

7.1 vivakṣā Dependent Operations

The application of many kāraka rules depends on the speaker's intention (*vivakṣā*) (Cardona 2004). In the example स्थाल्या पचति, स्थाली should be termed अधिकरण (locus) as it is the आधार (आधारोऽधिकरणम्), but it is termed करण by rule साधकतमं करणम् because the speaker thinks it is the most instrumental (साधकतम (प्रकृष्ट उपकारक) and therefore prefers instrumental case and hence the tṛtīyā is conditioned by कर्तृकरणयोस्तृतीया (2.3.18). कर्मणा यमभिप्रैति स सम्प्रदानम् prescribes सम्प्रदान for the receiver of a gift, but the vārtika अशिष्टव्यवहारे दाणः प्रयोगे चतुर्थ्यर्थे तृतीया prohibits it if the gift was intended for deriving some benefit (sexual favor in this case).

7.2 vārtikas

The vārtikas provide additional information on Paninian rules. They often introduce new conditions for the rules to apply as such, extend the rule application to new domains, or delimit them. This adds to the complexities because many of the Paninian rules have to be implemented taking vārtikas into account. For example, नी–वह्योर्न (नाययति वाहयति वा भारं भृत्येन) allows karaṇa if the verb is नी or वह्. It thus limits गतिबुद्धिप्रत्यवसानार्थ शब्दकर्माकर्मकाणामणि कर्ता स णौ which allows karma.[1] In another instance, the vārtikas can be seen as limiting themselves as follows:

नियन्तृकर्तृकस्य वहेरनिषेधः (if the kartā is sārathi or any of its synonyms then नी–वह्योर्न does not apply → वाहयति रथं वाहान् सूतः (karma by Pāṇini's गतिबुद्धि… sūtra)

7.3 Semantic Conditions

Semantic conditions such as those mentioned below are hard to implement. Some possible solutions have been provided in section 8.

स्वातंत्र्य

The functional property of a sentential constituent being capable of carrying out actions on its own is hard to implement using a computer algorithm. Possible solution could be building extensive ontological trees and compatibility maps for verbs and their arguments.

ईप्सित/ईप्सिततम

Desirable and less or more desirable objects for agents cannot be implemented without domain knowledge. The fact that poison is generally not going to be a more desirable object compared to rice is a well known fact. However, engineering this small piece of knowledge in computer is hard. We may have to build domain dictionaries for what is generally believed to be good and bad for members of different sets. Even if such common-sense world knowledge has been built in the machine, there will be idiosyncracies (or *vivakṣā*).

साधकतम

The argument here is more or less similar to that mentioned above for desirable/ more/less desirable. The vivakṣā will however play a larger role here than in the previous case.

---

[1] The sūtra (1.4.52) provides that the one independent in the action denoted by the base root be termed *karman* in relation to the action denoted by the causative root. By negating the sūtra in the cases of the two roots in question, the vārtika allows the one independent in the action to be termed *kartā* by 1.4.54 thereby conditioning the third triplet of nominal terminations by 2.3.18. There is no question of one independent in the action being termed *karaṇa* (Editor's note).

अभिप्रैति (to be अभिमुख - approach someone to give a gift)

Implementation of this will require ākāṅkṣā/yogyatā charts for verbs of giving. To be more specific, we will have to state which of such verbs are compatible with which gifts to what kinds of recipients.

ध्रुव (fixed point) अपाय (path of separation)

ध्रुव or fixed points of departure can be only formalized by encoding world knowledge. Sometimes, there may be confusion in deciding whether something is a ध्रुव or an आधार as in the example धावतः अश्वात् पतति, is अश्व an आधार or ध्रुव ?

आधारोऽधिकरणम् (आधारः किम् ?)

Similarly, what can be a matching आधार for a matching person or thing will be hard to encode in the machine. As in the previous cases massive knowledge bases of compatible objects will have to be built.

8 Some General Solutions

Besides building ontological knowledge bases with compatibility charts, we will have to develop several dictionaries and databases as follows:

- Build a database of all the kāraka rules, special conditions, exceptions, vārtikas, examples, etc. The authors have developed such a database
- Synonymy and equivalence lexicon like the *Amarakośa*
  - Rules having 'in the sense of' (वाची / अर्थक) etc can be handled by a synonymy lexicon like the *Amarakośa* (http://sanskrit.jnu.ac.in/amara/index.jsp)
- Using wordnet and other lexical resources for Sanskrit like those developed by Kulkarni (2008)
- Semantic classification of verbs according to kāraka to handle abundant cases like 'verbs in the sense of …' and the like. The *Dhātupāṭha* gives a semantic grouping of Sanskrit verbs as can be seen at the following location:
  - http://sanskrit.jnu.ac.in/tinanta/tinanta.jsp
- Another option could be deciding on a set of primitives for each of these concepts and a unique forward (building) mechanism for building larger concepts from basic ones and a backward (reducing) mechanism for reducing compound concepts into unitary ones. We can model this solution on the lines of Jackendoff and Dorr and use them for Sanskrit verb classes incorporating kāraka information as well.

9 Model for Kāraka Identification

A tentative model for kāraka marking has been done and partial implementation has been done at http://sanskrit.jnu.ac.in/karaka/analyzer.jsp. Of the eight modules, six have been developed. Modules 5, 6, and 7 are under development as of now.

[1] VERB ID
[2] VERB ANALYSIS
[3] NON—VERB ID
[4] SUBANTA ANALYSIS
***[5] AKANKSHA CHECK**
***[6] KARAKA RULES**
***[7] SPECIAL CONDITIONS**
[8] KARAKA ASSIGNMENT

The module names prefixed with '*' are under development currently.

Sample example - devadattaḥ odanaṁ pacati

[1]
→ devadattaḥ odanaṁ pacati [VERB]

[2]
→devadattaḥ odanaṁ pacati
[ḍupacaṣ>dvikarmaka>bhvādigaṇa>kartṛvācya]

[3]
→ devadattaḥ[SUBANTA] odanaṁ[SUBANTA] pacati
ḍupacaṣ>dvikarmaka>bhvādigaṇa>kartṛvācya]

[4]
→ devadattaḥ_1.1 odanaṁ_2.1/2.1 pacati
ḍupacaṣ>dvikarmaka>bhvādigaṇa>kartṛvācya]

[5]
[6]
[7]

[8]
→ devadattaḥ_1.1[kartṛ (svatantraḥ…)] odanaṁ_2.1[karman (karturīpsitatam…)] pacati [ḍupacaṣ>dvikarmaka>bhvādigaṇa>kartṛvācya]

10 Conclusion

The authors in this paper have tried to present a case for deeper analysis of semantic conditions in rule writing of Pāṇini and have highlighted the fact that a complete implementation of the system of Pāṇini will not be possible without formalizing all kāraka rules as per the semantic conditions put forth by Pāṇini. This may also

necessitate a proper understanding of vivakṣā and its formalization. The fact that this is a difficult task given the status of computing technologies available to us has also been highlighted by the authors. Some long term solutions have also been suggested. The authors have also put forth a kāraka analyzer system consisting of several modules many of which have been developed.

References

Bhatta, V.P.: Epistemology, logic and grammar, in the analysis of sentence-meaning. South Asia Books (1992)

Cardona, G.: Pāṇini's syntactic categories. Journal of the Oriental Institute, Baroda (JOIB) 16, 201–215 (1967)

Cardona, G.: Pāṇini: his work and its tradition, vol. 1. Motilal Banarasidas, Delhi (1988)

Cardona, G.: Some questions on Pāṇini's derivational system. In: Proceedings of International Conference on Speech and Language Technology, vol. 1, Tata McGraw Hill, New York (2004)

Ghildiyal, S.: Vaiyākaraṇa Siddhānta Kaumudī - Hindi vyākhyā sahita vibhaktyartha (kāraka prakaraṇa). Motilal Banarasidas, Delhi (1962)

Ishvarchand: Aṣṭādhyāyī, vol. 1-2. Chaukhamba Surabharati Prakashan, Delhi (2004)

Jha, G.N.: The System of Pāṇini. Language in India 4.2 (2004)

Jha, G.N., et al.: Towards a Computational Analysis System for Sanskrit. In: Proc. of First National Symposium on Modeling and Shallow Parsing of Indian Languages, pp. 25–34. Indian Institute of Technology, Bombay (2006)

Mishra, S.K., Jha, G.N.: Identifying Verb Inflections In Sanskrit Morphology. In: Proc. of SIMPLE 2004, IIT, Kharagpur, pp. 79–81 (2004)

Mishra, V.A.: Paribhāṣenduśekhara. Chaukhamba Surabharati Prakashan, Delhi (1997)

Mishra, S.K., Jha, G.N.: Sanskrit Kāraka Analyzer For Machine Translation. In: SPLASH proc. of iSTRANS, pp. 224–225. Tata McGraw-Hill, New Delhi (2004)

Mishra, S. (ed.): Kāśikā of Vāmana And Jayāditya. Chaukhamba Sanskrit Sansthan, Varanasi (1996)

Nayar, V.R.P.: Logical Structure of Kārakas. In: Aspects of Pāṇinian Semantics, pp. 99–105. Sahitya Academy, New Delhi (2003)

Rajendran, C.: Aspects of Pāṇinian Semantics. Sahitya Academy, New Delhi (2003)

Ramanujam: Semantic problems in the computation of Pāṇinian rules. In: Aspects of Pāṇinian Semantics. Sahitya Academy, New Delhi (2003)

Sharma, R.N.: The Aṣṭādhyāyī of Pāṇini. Munshiram Manoharlal Publishers Pvt. Ltd., Delhi (2003)

Whitney, W.D.: A Sanskrit grammar: including both the classical language, and the older dialects, of Veda and Brahmana. Breitkopf & Härtel (1896); History of Sanskrit grammar. Sanjay Prakashan, Delhi (reprint, 2002)

From Pāṇinian Sandhi to Finite State Calculus

Malcolm D. Hyman*

Max Planck Institute for the History of Science
Boltzmannstr. 22, D-14195 Berlin, Germany
hyman@mpiwg-berlin.mpg.de

Abstract. The most authoritative description of the morphophonemic rules that apply at word boundaries (external sandhi) in Sanskrit is by the great grammarian Pāṇini (fl. 5th c. B. C. E.). These rules are stated formally in Pāṇini's grammar, the *Aṣṭādhyāyī* 'group of eight chapters'. The present paper summarizes Pāṇini's handling of sandhi, his notational conventions, and formal properties of his theory. An XML vocabulary for expressing Pāṇini's rules is introduced and the application to morphophonemic rules demonstrated. Although Pāṇini's notation potentially exceeds a finite state grammar in power, individual rules do not rewrite their own output, and thus they may be automatically translated into a rule cascade from which a finite state transducer can be compiled.

1 Sandhi in Sanskrit

Sanskrit possesses a set of morphophonemic rules (both obligatory and optional) that apply at morpheme and word boundaries (the latter are also termed *pada boundaries*). The former are called *internal sandhi* (< *saṁdhi* 'putting together'); the latter, *external sandhi.* This paper only considers external sandhi. Sandhi rules involve processes such as assimilation and vowel coalescence. Some examples of external sandhi are: *na asti* > *nāsti* 'is not', *tat ca* > *tac ca* 'and this', *etat hi* > *etad dhi* 'for this', *devas api* > *devo 'pi* 'also a god'.[1]

2 Sandhi in Pāṇini's Grammar

Pāṇini's *Aṣṭādhyāyī* is a complete grammar of Sanskrit, covering phonology, morphology, syntax, semantics, and even pragmatics. It contains about 4000 rules (termed *sūtra,* literally 'thread'), divided between eight chapters (termed *adhyāya*). Conciseness (*lāghava*) is a fundamental principle in Pāṇini's formulation of carefully interrelated rules [1]. Rules are either *operational* (i. e. they

* This work has been supported by NSF grant IIS-0535207. Any opinions, findings, and conclusions or recommendations expressed are those of the author and do not necessarily reflect the views of the National Science Foundation. The paper has benefited from comments by Peter M. Scharf and by four anonymous referees.

[1] The symbol ⟨'⟩ (*avagraha*) does not represent a phoneme but is an orthographic convention to indicate the prodelision of an initial *a*-.

G. Huet, A. Kulkarni, and P. Scharf (Eds.): Sanskrit CL 2007/2008, LNAI 5402, pp. 253–265, 2009.

specify a particular linguistic operation, or *kārya*) or *interpretive* (i. e. they define the scope of operational rules).[2] Rules may be obligatory or optional.

A brief review of some well-known aspects of Pāṇini's grammar is in order. The operational rules relevant to sandhi specify that a substituend (*sthānin*) is replaced by a substituens (*ādeśa*) in a given context [3, 308]. Rules are written using metalinguistic case conventions, so that the substituend is marked as genitive, the substituens as nominative, the left context as ablative (*tasmāt*), and the right context as locative (*tasmin*). For instance:

8.4.62	*jhayo*	*ho*	*'nyatarasyām*
	jhaY-ABL	*h*-GEN	optionally

This rule specifies that (optionally) a homogeneous sound replaces *h* when preceded by a sound termed *jhaY* — i. e. an oral stop [4, 783–784]. Pāṇini uses abbreviatory labels (termed *pratyāhāra*) to describe phonological classes. These labels are interpreted in the context of an ancillary text of the *Aṣṭādhyāyī,* the *Śivasūtras,* which enumerate a catalog of sounds (*varṇasamāmnāya*) in fourteen classes [5, 6]:

1. a i u Ṇ
2. r̥ l̥ K
3. e o Ṅ
4. ai au C
5. h y v r Ṭ
6. l Ṇ
7. ñ m ṅ ṇ n M
8. jh bh Ñ
9. gh ḍh dh Ṣ
10. j b g ḍ d Ś
11. kh ph ch ṭh th c ṭ t V
12. k p Y
13. ś ṣ s R
14. h L

The final items (indicated here by capital letters) are markers termed *it* 'indicatory sound' and are not considered to belong to the class. A *pratyāhāra* formed from a sound and an *it* denotes all sounds in the sequence beginning with the specified sound and ending with the last sound before the *it.* Thus *jhaY* denotes the class of all sounds from *jh* through *p* (before the *it Y*): *jh, bh, gh, ḍh, dh, j, b, g, ḍ, d, kh, ph, ch, ṭh, th, c, ṭ, t, k, p* (i. e. all oral stops).[3] Sūtra 8.4.62, as printed above, is by itself both elliptic and uninterpretable. Ellipses in sūtras are completed by supplying elements that occur in earlier sūtras; the device by which omitted elements can be inferred from preceding sūtras is termed *anuvr̥tti* 'recurrence' [2, 60]. It will be noticed that no substituens is specified in 8.4.62; the substituens *savarnaḥ* 'homogeneous sound-NOM' [6] is supplied by *anuvr̥tti* from sūtra 8.4.58 *anusvārasya yayi parasavarṇaḥ.* Still, the device of *anuvr̥tti* is insufficient to specify the exact sound that must be introduced as a substituens. It is here that interpretive rules play a role. Sūtra 8.4.62 must be interpreted

[2] The traditional classification of rules is more fine-grained and comprises *saṁjñā* (technical terms), *paribhāṣā* (interpretive rules), *vidhi* (operational rules), *niyama* (restriction rules), *pratiṣedha* (negation rules), *atideśa* (extension rules), *vibhāṣā* (optional rules), *nipātana* (ad hoc rules), *adhikāra* (heading rules) [2, 89].

[3] The vowel *a* added after a consonant makes the *pratyāhāra* pronounceable.

in the light of the *paribhāṣā* 'interpretive rule' 1.1.50 *sthāne 'ntaratamaḥ*, which specifies that a substituens must be maximally similar (sc. in articulatory place and manner) to the substituend [2, 126]. Thus the substituens in 8.4.62 will always be aspirated, since *h* (the substituend) is aspirated: e. g. *vāg hasati* (< *vāk hasati* 'a voice laughs' by 8.2.39) > *vāgghasati.*[4]

A second example further illustrates the principles already discussed:

8.4.63 *śaś* *cho* *'ṭi*
ś-GEN *ch*-NOM *aṬ*-LOC

Here *jhayaḥ* '*jhaY*-ABL' and *anyatarasyām* 'optionally' are supplied by *anuvr̥tti* from the preceding sūtra (8.4.62). The rule specifies that (optionally) *ś* is replaced by *ch* when it is preceded by an oral stop (*jhaY*) and followed by a vowel or semivowel (*aṬ*).

Rules specific to external sandhi are found in the third quarter (*pāda*) of the eighth *adhyāya.* A number of rules are common to both internal and external sandhi, and rules relevant for external sandhi are also found in the first *pāda* of the sixth *adhyāya* and the fourth *pāda* of the eighth *adhyāya.*

3 An XML Encoding for Pāṇinian Rules

An encoding based on Extensible Markup Language (XML) has been chosen for expressing Pāṇinian rules in machine-readable form. The XML vocabulary contains an element `<rule>`, with required attributes `source` (the substituend) and `target` (the substituens) and optional attributes `lcontext` (the left context) and `rcontext` (the right context). The values of `source`, `lcontext`, and `rcontext` are specified as Perl-compatible regular expressions (PCREs) [7].[5] Sanskrit sounds are indicated in an encoding known as SLP1 (Sanskrit Library Phonetic 1) [8]. An encoding such as Unicode is not used, since Unicode represents *written characters* rather than *speech sounds* [9]. The SLP1 encoding facilitates linguistic processing by representing each Sanskrit sound with a single symbol (see Fig. 1).[6] The use of Unicode would be undesirable here, since (1) there would not be a one-to-one correspondence between character and sound, and (2) Sanskrit is commonly written in a number of different scripts (Devanāgarī, Tamil, Roman, etc.).

The following is the XML representation of sūtra 8.3.23 *mo 'nusvāraḥ,* which specifies that a *pada*-final *m* is replaced by the nasal sound *anusvāra* (*ṁ*) when followed by a consonant [4, 628]:

[4] Note that Sanskrit *h* represents a voiced glottal fricative [ɦ].

[5] So-called "regular expressions" in programming languages such as Perl include extended features such as pattern memory that exceed the power of regular languages (see Sect. 4); for example, it is possible to write a PCRE that matches the $\alpha\alpha$ language, that is, the language of all reduplicated strings. Thus PCREs are *not,* in the formal sense, regular expressions at all.

[6] Figure 1 shows only the basic phonetic segments in the SLP1 encoding. The entire encoding is documented in [8].

```
<rule source="m" target="M"
      rcontext="[@(wb)][@(hal)]"
      ref="A.8.3.23"/>
```

This rule employs a syntactic extension used for *macros*. The expression `@(name)` is replaced by the value of a defined macro `name`. Macros are defined with an XML element `<macro>` and may be defined recursively. Macro expansion is performed immediately after parsing the XML file, before any further processing. Here the `rcontext` attribute references two macros: `@(wb)` is expanded to characters that indicate a word boundary, and `@(hal)` is expanded to the characters representing the sounds of the *pratyāhāra haL* (i. e. all consonants). Macros are a syntactic convenience that allows rules to be easier to read (and closer to Pāṇini's original formulation); one might equally spell out in full all characters representing sounds in a phonological class. The square brackets belong to the PCRE syntax and indicate that any character contained between them should be matched; that is, `[abc]` matches an `a`, `b`, or `c`.

अ a **a**	आ ā **A**	इ i **i**	ई ī **I**	उ u **u**	ऊ ū **U**

ऋ ṛ **f**	ॠ ṝ **F**	ऌ ḷ **x**	ॡ ḹ **X**
ए e **e**	ऐ ai **E**	ओ o **o**	औ au **O**

क् k **k**	ख् kh **K**	ग् g **g**	घ् gh **G**	ङ् ṅ **N**
च् c **c**	छ् ch **C**	ज् j **j**	झ् jh **J**	ञ् ñ **Y**
ट् ṭ **w**	ठ् ṭh **W**	ड् ḍ **q**	ढ् ḍh **Q**	ण् ṇ **R**
		ळ् ḷ **L**	ळ्ह् ḷh **|**	
त् t **t**	थ् th **T**	द् d **d**	ध् dh **D**	न् n **n**
प् p **p**	फ् ph **P**	ब् b **b**	भ् bh **B**	म् m **m**

य् y **y**	र् r **r**	ल् l **l**	व् v **v**
श् ś **S**	ष् ṣ **z**	स् s **s**	ह् h **h**
ः ḥ **H**	ᳵ ẖ **Z**	ᳶ ḫ **V**	ं ṁ **M**

Fig. 1. Basic phonetic segments in the SLP1 encoding

In the `target` two additional syntactic facilities allow for rules to be expressed in a way that is close to Pāṇini's formulation. These facilities are termed *mappings* and *functions.* The following rule illustrates a mapping:

```
<rule source="h"
      target="%(voicedaspirate($1))"
      lcontext="([@(Jay)])[@(wb)]"
      optional="yes"
      ref="A.8.4.62"/>
```

This sūtra has been discussed earlier. The left context matches a *jhaY* followed by a word boundary. The parentheses (part of PCRE syntax) specify that the contents (the *jhaY*) be stored in pattern memory. Strings stored in pattern memory are available for subsequent reference: the pattern matched by the first parenthesized group may be recalled with `$1`; the second, with `$2`; etc. The pattern memory variable `$1` is referenced in the `target` attribute. The rule specifies that an *h*, when preceded by a *pada*-final *jhaY*, is replaced by the result of performing the `voicedaspirate` mapping on the matched *jhaY.* The mapping syntax takes the form `%(`***`name`***`(`***`input`***`))`, where ***`name`*** is the name of a mapping, and ***`input`*** is the input symbol for the mapping. A mapping is defined with a `<mapping>` element, which has as children one or more `<map>` elements. The mapping `voicedaspirate` is defined thus:

```
<mapping name="voicedaspirate">
  <map from="@(jaS)" to="@(Jaz)"/>
</mapping>
```

This mapping translates the voiced oral stops denoted by the *pratyāhāra jaŚ* (*j, b, g, ḍ, d*) to the equivalent aspirated voiced oral stops denoted by the *pratyāhāra jhaṢ* (*jh, bh, gh, ḍh, dh*). In the case that the input symbol to a mapping is not contained in any `from`, the mapping is equivalent to the identity mapping.

Sūtra 6.1.87 *ād guṇaḥ* illustrates the use of a function:

```
<rule source="[@(a)][@(wb)]([@(ik)])"
      target="!(gunate($1))"
      ref="A.6.1.87"/>
```

The `source` matches either short *a* or long *ā* (by the definition of the macro `a`), a word boundary, and then the *pratyāhāra iK* (a simple vowel other than *a* or *ā*). The macro `ik` is defined:

```
<macro name="ik"
       value="@(i)@(u)@(f)@(x)"
       ref="A.1.1.71"/>
```

and depends on the macro definitions:

```
<macro name="i" value="iI"
       ref="A.1.1.69"/>
```

```
<macro name="u" value="uU"
       ref="A.1.1.69"/>
<macro name="f" value="fF"
       ref="A.1.1.69"/>
<macro name="x" value="xX"
       ref="A.1.1.69"/>
```

The *iK* is stored in pattern memory, and the substituend (*a-varṇa*) is replaced by the output of calling the function `gunate` on the stored *iK.* A function is defined with an element `<function>`, which has as children one or more `<rule>` elements. These are context-free rules that typically make use of mappings in the `target`. Thus `gunate` is defined:

```
<function name="gunate">
  <rule source="[@(a)@(i)@(u)]"
        target="%(guna($1))"/>
  <rule source="[@(f)@(x)]"
        target="%(guna($1))
                %(semivowel($1))"/>
</function>
```

Two mappings are invoked here:

```
<mapping name="guna"
         ref="A.1.1.2">
  <map from="@(a)" to="a"/>
  <map from="@(i)" to="e"/>
  <map from="@(u)" to="o"/>
  <map from="@(f)" to="a"/>
  <map from="@(x)" to="a"/>
</mapping>

<mapping name="semivowel"
         ref="A.6.1.77">
  <map from="@(i)" to="y"/>
  <map from="@(u)" to="v"/>
  <map from="@(f)" to="r"/>
  <map from="@(x)" to="l"/>
</mapping>
```

The mapping `guna` maps simple vowels such as *a-varṇa* (which includes short *a* and long *ā*) to their *guṇa* equivalent. The term *guṇa* is defined by the technical rule (*saṁjñā*) [2, 102] 1.1.2 *adeṅ guṇaḥ* '*a* and *eṄ* [are] *guṇa*' (the *pratyāhāra eṄ* = {*e, o*}). The mapping `semivowel` maps those vowels that possess homorganic semivowels to the corresponding semivowels. The function `gunate`, if the input is *a-, i-,* or *u-varṇa,* outputs the corresponding *guṇa* vowel; if the input is *r̥-* or *l̥-varṇa,* it outputs the corresponding *guṇa* vowel (in this case, *a*) concatenated with the homorganic semivowel (either *r* or *l*). The domain of `gunate` is {*a, ā, i, ī, u, ū, r̥, r̥̄, l̥, l̥̄*} and its range is {*a, e, o, ar, al*}.

4 Regular Languages and Regular Relations

First, it is necessary to define a *regular language.* Let Σ denote a finite alphabet and Σ^ϵ denote $\Sigma \cup \{\epsilon\}$ (where ϵ is the empty string). $\{e\}$ is a regular language where $e \in \Sigma^\epsilon$, and the empty language $\emptyset$ is a regular language. Given that L_1, L_2, and L are regular languages, additional regular languages may be defined by three operations (under which they are closed): concatenation ($L_1 \cdot L_2 = \{xy|x \in L_1, y \in L_2$), union ($L_1 \cup L_2$), and Kleene closure ($L^* = \cup_{i=0}^{\infty} L^i$) [10, 338].

Regular relations are defined in the same fashion. An n-relation is a set whose members are ordered n-tuples [11, 20]. Then $\{e\}$ is a regular n-relation where $e \in \Sigma^\epsilon \times \ldots \times \Sigma^\epsilon$ ($\emptyset$ is also a regular n-relation). Given that R_1, R_2, and R are regular n-relations, additional regular n-relations may be defined by three operations (under which they are closed): n-way concatenation ($R_1 \cdot R_2 = \{xy|x \in R_1, y \in R_2$), union ($R_1 \cup R_2$), and n-way Kleene closure ($R^* = \cup_{i=0}^{\infty} R^i$).

5 Finite State Automata and Regular Grammars

A *finite state automaton* is a mathematical model that corresponds to a regular language or regular relation [11, 44]. A simple finite state automaton corresponds to a regular language and is a quintuple $\langle S, \Sigma, \delta, s_0, F \rangle$, where S is a finite set of states, Σ the alphabet of the automaton, δ is a transition function that maps $S \times \Sigma^\epsilon$ to 2^S, $s_0 \in S$ is a single initial state, and $F \subseteq S$ is a set of final states [12, 114]. A finite state automaton that corresponds to a regular relation is termed a *finite state transducer* (FST) and can be defined as a quintuple $\langle S, \Sigma \times \ldots \times \Sigma, \delta, s_0, F \rangle$, with δ being a transition function that maps $S \times \Sigma^\epsilon \times \ldots \times \Sigma^\epsilon$ to 2^S [10, 340].

A regular (or finite) grammar describes a regular language and is equivalent to a finite state automaton. In a regular grammar, all production rules have a single non-terminal on the left-hand side, and either a single terminal or a combination of a single non-terminal and a single terminal on the right-hand side. That is, all rules are of the form $A \rightarrow a$, $A \rightarrow aB$ (for a right regular grammar), or $A \rightarrow Ba$ (for a left regular grammar), where A and B are single non-terminals, and a is a single terminal (or ϵ). A regular grammar is the least powerful type of grammar in the Chomsky hierarchy (see Fig. 2) [13]. A context free grammar describes a context free language, a context sensitive grammar describes a context sensitive language, and an unrestricted grammar describes a recursively enumerable language. The hierarchy is characterized by proper inclusion, so that every regular language is context free, every context free language is context sensitive, etc. (but not every context free language is regular, etc.). A regular grammar cannot describe a context free language such as $\{a^n b^n | 1 \leq n\}$, which consists of the strings $\{ab, aabb, aaabbb, aaaabbbb \ldots\}$ [10, 346].

Since Pāṇinian rules for external sandhi can be modeled using a finite grammar, it is highly desirable to provide a finite state implementation, which is computationally efficient. Finite state machines are closed under composition, and thus sandhi operations may be composed with other finite state operations to yield a single network.

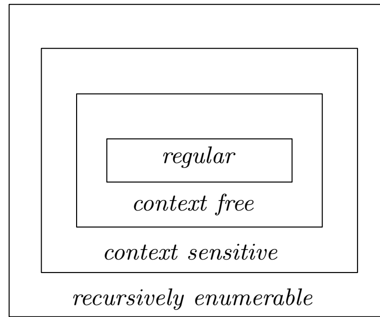


Fig. 2. The Chomsky hierarchy of languages (after [14, 3])

6 From Rewrite Rules to Regular Grammars

A string rewriting system is a system that can transform a given string by means of rewrite rules. A rewrite rule specifies that a substring $x_1 \ldots x_n$ is replaced by a substring $y_1 \ldots y_m$: $x_1 \ldots x_n \rightarrow y_1 \ldots y_m$, where $x_i, y_i \in \Sigma$ (and Σ is a finite alphabet). Rewrite rules used in phonology have the general form

$$\phi \rightarrow \psi / \lambda \,\underline{\quad}\, \rho \,. \tag{1}$$

Such a rule specifies that ϕ is replaced by ψ when it is preceded by λ and followed by ρ. Traditionally, the phonological component of a natural-language grammar has been conceived of as an ordered series of rewrite rules (sometimes termed a *cascade*) $W_1, \ldots, W_n$ [15, 20]. Most phonological rules, however, can be expressed as regular relations [11, 33]. But if a rewrite rule is allowed to rewrite a substring introduced by an earlier application of the same rule, the rewrite rule exceeds the power of regular relations [10, 346]. In practice, few such rules are posited by phonologists. Although certain marginal morphophonological phenomena (such as arbitrary center embedding and unlimited reduplication) exceed finite state power, the vast majority of (morpho)phonological processes may be captured by regular relations [11, 419].

Since phonological rewrite rules normally (with the provisos discussed above) correspond to regular relations, they may be modeled as FSTs. FSTs are closed under composition; thus if T_1 and T_2 are FSTs, application of the composed transducer $T_1 \circ T_2$ to a string S is equivalent to applying T_1 to S and applying T_2 to the output of T_1:

$$T_1 \circ T_2(S) = T_2(T_1(S)) \,. \tag{2}$$

So if a cascade of rewrite rules $W_1, \ldots, W_n$ can be expressed as a series of FSTs $T_1, \ldots, T_n$, there is a single FST T that is a composition $T_1 \circ \ldots \circ T_n$ and is equivalent to the cascade $W_1, \ldots, W_n$ [10, 364]. $G = T_1 \circ \ldots \circ T_n$ constitutes a regular grammar. Efficient algorithms are known for compiling a cascade of rewrite rules into an FST [16].

7 An FST for Pāṇinian Sandhi

The XML formalism for expressing Pāṇinian rules in Sect. 3 contains a number of devices; it is not immediately evident how rules employing these devices might be compiled into an FST. A rule compiler, however, is described here that translates Pāṇinian rules expressed in the XML formalism into rewrite rules that can be automatically compiled into an FST using standard algorithms.

Table 1 shows the Pāṇinian derivation of the string *devo 'pi* < *devas api* 'also a god'. Sūtra 8.2.66 replaces a *pada*-final *s* with *rU* (here symbolized by `$`). Sūtra 6.1.113 replaces *pada*-final *rU* with *u* preceded by a boundary marker (`#`) when the next *pada* begins with *a*. Sūtra 6.1.87 has been discussed above. Sūtra 6.1.109 replaces *pada*-initial *a* with *avagraha* (represented by `'` in SLP1) when the previous *pada* ends in the *pratyāhāra eṄ* (i. e. *e* or *o*).[7]

Table 1. Derivation of *devo 'pi*

FORM	SŪTRA	TEXT
`devas api`		
`deva$ api`	8.2.66	*sasajuṣo ruḥ*
`deva#u api`	6.1.113	*ato ror aplutād aplute*
`dev!(gunate(u)) api`	6.1.87	*ād guṇaḥ*
`devo 'pi`	6.1.109	*eṅaḥ padāntād ati*

Although the PCREs in the XML format exceed the power of regular relations, and the implementation of mappings and functions is not obvious in a regular grammar, the rule compiler mentioned above is able to produce a cascade of rewrite rules that may be efficiently compiled into an FST. A consequence of the rule compilation strategy is that a single Pāṇinian rule may be represented as several rewrite rules. Where a rule makes use of pattern memory, which can contain n possible values, the rule is expanded into n rewrite rules. Mappings and functions are automatically applied where they occur in ψ (the substituens).

Table 2 illustrates a cascade representing the four sūtras exemplified in Table 1. Here the notation $(x|y|z)$ expresses alternation; the rule matches either x, or y, or z. The symbol ␣ represents a space character; the symbol `#` represents a boundary that has been inserted by a rule.

These rules may be efficiently compiled into FSTs. An FST encoding the final rule is shown in Fig. 3. In this transducer, s_0 is the initial state and doubly-circled states are the members of F, the set of final states. The graphical representation of the FST has been simplified, so that symbols that are accepted on transition from s_i to s_j share an arc between s_i and s_j (in this case, the symbols are separated by commas). The notation $x : y$ indicates the symbols on the upper and lower tapes of the transducer, respectively. If the transducer reads input

[7] Although *avagraha* is not a phoneme, it may be conceived of linguistically as a "trace" left after the deletion of *a-*. For this reason, it is represented in the SLP1 phonetic coding.

Table 2. Rule cascade for deriving *devo 'pi*

```
s → $ / ____ (␣ | #)
$ → #u / a ____ (␣ |#)a
(a|A)(␣ | #)(a|A) → a
(a|A)(␣ | #)(i|I) → e
(a|A)(␣ | #)(u|U) → o
(a|A)(␣ | #)(f|F) → ar
(a|A)(␣ | #)(x|X) → al
a → ' / (e|o)(␣ | #) ____
```

from the upper tape and writes output to the lower tape, $x : y$ indicates that if x is read on the upper tape, y is written on the lower tape. If the symbols on the upper and lower tapes are the same, a shorthand notation is used; thus x is equivalent to $x : x$. The special symbol ? indicates that *any* symbol is matched on either the upper or lower tape.

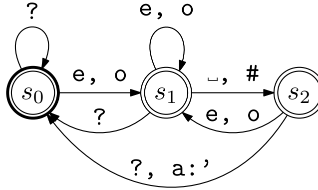


Fig. 3. An FST for sūtra 6.1.109

Since it is possible to translate each rule in Table 2 into an FST, and FSTs are closed under composition, it is possible to compose a single FST that implements the portion of Pāṇini's sandhi represented by the rules in Table 2. A compiler developed by the author will be capable of compiling the entirety of Pāṇini's sandhi rules into a single FST. The FST is then converted to Java code using a toolkit written by the author. Compilation of the generated source code yields binary code that may be run portably using any Java Virtual Machine (JVM) [17]. Alternatively, for improved performance, the Java code may be compiled into native machine code using the GNU Compiler for Java (gcj) [18].

8 Implications

While one of the guiding principles of Pāṇini's grammar is conciseness (*lāghava*), a computational implementation poses other demands, such as tractability and efficiency. Pāṇini's rules are formulated in terms of classes based on distinctive features and must be construed with the aid of 'interpretive' (*paribhāṣā*) rules, technical terms (*saṁjñā*), and other theoretical apparatus.

However desirable the mathematical properties of a regular grammar may be, a grammar stated in such terms is at odds with Pāṇini's principles. Rather, it

hearkens back to the *vikāra* system employed by earlier linguistic thinkers, in which individual segments are the target of specific rules [3, 311].

By way of contrast, Pāṇini states a rule economically in terms of the sound classes enumerated in the *Śivasūtras*. Thus 8.4.41 *ṣṭunā ṣṭuḥ* specifies the retroflexion of an *s* or dental stop either (1) before a *pada*-final -*ṣ* or (2) *pada*-finally before a *ṭU* (i. e. a retroflex stop).

With a little less economy, we can represent Pāṇini's rule by two rules in XML:

```
<rule source="[s@(tu)]"
      target="%(retroflex($1))"
      lcontext="z[@(wb)]"
      ref="A.8.4.41"/>

<rule source="[s@(tu)]"
      target="%(retroflex($1)"
      rcontext="[@(wb)][@(wu)]"
      ref="A.8.4.41"/>
```

The *pratyāhāra tU* stands for the dental stop series {*t, th, d, dh, n*}. We apply the retroflex mapping:

```
<mapping name="retroflex">
  <map from="s" to "z"/>
  <map from="@(tu)" to "@(wu)"/>
</mapping>
```

The effect is to change a single phonological feature across an entire class; sounds with a dental place of articulation (*dantya*) are replaced by sounds with a retroflex articulation (*mūrdhanya*). In specifying such a replacement, Pāṇini makes use of the principle of *sāvarṇya* 'homogeneity of sounds'. So the substituend chosen is that closest to the original — with respect to voice (*ghoṣavat / aghoṣa*), aspiration (*mahāprāṇa / alpaprāṇa*), and nasality (*sānunāsika / niranunāsika*). Thus *t* → *ṭ, th* → *ṭh, d* → *ḍ,* and so on; yet Pāṇini does not need explicitly (and repetitively) to specify the exact segments substituted. In this way, the *Aṣṭādhyāyī* avoids the bias ("segmentalism") that places the linear segment at the center of phonological theory — a bias from which contemporary linguistics is beginning to distance itself [19].

The finite state approach discussed in this paper, however, is limited to describing the relations between strings (sequences of segments). As far as the computational model is concerned, individual symbols are atomic, and no class relations between the symbols exist. The goal in this study has been to develop an intermediate representational structure, based on XML, that can faithfully encode some of the linguistically significant aspects of a portion of Pāṇini's grammar (the rules involved in external sandhi) and at the same time can be automatically translated into an efficient computational implementation.

References

1. Smith, H.: Brevity in Pāṇini. Journal of Indian Philosophy 20, 133–147 (1992)
2. Sharma, R.N.: The Aṣṭādhyāyī of Pāṇini. Introduction to the Aṣṭādhyāyī as a Grammatical Device, vol. 1. Munshiram Manoharlal, New Delhi (1987)
3. Cardona, G.: On Translating and Formalizing Pāṇinian Rules. Journal of the Oriental Institute, Baroda 14, 306–314 (1965)
4. Sharma, R.N.: The Aṣṭādhyāyī of Pāṇini. English Translation of Adhyāyas Seven and Eight with Sanskrit Text, Transliteration, Word-Boundary, Anuvṛtti, Vṛtti, Explanatory Notes, Derivational History of Examples, and Indices, vol. 6. Munshiram Manoharlal, New Delhi (2003)
5. Cardona, G.: Studies in Indian Grammarians: I. The Method of Description Reflected in the Śivasūtras. Transactions of the American Philosophical Society 59, 3–48 (1969)
6. Cardona, G.: On Pāṇini's Morphophonemic Principles. Language 41, 225–237 (1965)
7. Wall, L., Christiansen, T., Orwant, J.: Programming Perl., 3rd edn. O'Reilly, Sebastapol (2000)
8. Scharf, P.M., Hyman, M.D.: Linguistic issues in encoding Sanskrit. Brown University (2008) (unpublished manuscript)
9. Unicode Consortium: The Unicode Standard, Version 5.0. Addison-Wesley, Boston (2006)
10. Kaplan, R.M., Kay, M.: Regular models of phonological rule systems. Computational Linguistics 20, 332–378 (1994)
11. Beesley, K.R., Karttunen, L.: Finite State Morphology. CSLI, Stanford (2003)
12. Aho, A.V., Sethi, R., Ullman, J.D.: Compilers: Principles, Techniques, and Tools. Addison-Wesley, Reading (1988)
13. Chomsky, N.: Three models for the description of language. IEEE Transactions on Information Theory 2, 113–124 (1956)
14. Prusinkiewicz, P., Lindenmayer, A.: The Algorithmic Beauty of Plants. Springer, New York (1990)
15. Chomsky, N., Halle, M.: The Sound Pattern of English. MIT Press, Cambridge (1968)
16. Mohri, M., Sproat, R.: An efficient compiler for weighted rewrite rules. In: 34th Annual Meeting of the Association for Computational Linguistics, Santa Cruz, CA, ACL, pp. 231–238 (1996)
17. Lindholm, T., Yellin, F.: The Java™ Virtual Machine Specification, 2nd edn. Addison-Wesley, Reading (1999)
18. Free Software Foundation: The GNU compiler for the Java™ programming language, `http://gcc.gnu.org/java/`
19. Aronoff, M.: Segmentalism in linguistics: The alphabetic basis of phonological theory. In: Downing, P., Lima, S.D., Noonan, M. (eds.) The Linguistics of Literacy. Typological Studies in Language, vol. 21, pp. 71–82. John Benjamins, Amsterdam (1992)

Appendix: Core Rules for External Sandhi

The following is a list of modeled *vidhi* rules (8.3.4: *adhikāra*, 8.4.44: *pratiṣedha*) for external sandhi. No account is taken here of *pragr̥hya* rules that specify

certain sounds as exempt from sandhi (since these rules refer to morphosyntactic categories). Various optional rules are not listed.

6.1.73 *che ca*
6.1.74 *āṅmāṅośca*
6.1.132 *etattadoḥ sulopo 'koranañsamāse hali*
8.2.66 *sasajuṣo ruḥ*
8.2.68 *ahan*
6.1.113 *ato roraplutādaplute*
6.1.114 *haśi ca*
6.1.101 *akaḥ savarṇe dīrghaḥ*
6.1.88 *vr̥ddhireci*
6.1.87 *ādguṇaḥ*
6.1.77 *iko yaṇaci*
6.1.109 *eṅaḥ padāntādati*
6.1.78 *eco 'yavāyāvaḥ*
8.2.39 *jhalāṁ jaśo 'nte*
8.3.4 *anunāsikātparo 'nusvāraḥ*
8.3.7 *naśchavyapraśān*
8.3.14 *ro ri*
8.3.17 *bhobhagoaghoapūrvasya yo 'śi*
8.3.15 *kharavasānayorvisarjanīyaḥ*
8.3.19 *lopaḥ śākalyasya*
8.3.20 *oto gārgyasya*
8.3.23 *mo 'nusvāraḥ*
8.3.31 *śi tuk*
8.3.32 *ṅamo hrasvādaci ṅamuṇnityam*
8.3.34 *visarjanīyasya saḥ*
8.3.35 *śarpare visarjanīyaḥ*
8.3.40 *namaspurasorgatyoḥ*
8.4.40 *stoḥ ścunā ścuḥ*
8.4.44 *śāt*
8.4.41 *ṣṭunā ṣṭuḥ*
8.4.45 *yaro 'nunāsike 'nunāsiko vā*
8.4.53 *jhalāṁ jaśjhaśi*
8.4.55 *khari ca*
8.4.60 *torli*
8.4.62 *jhayo ho 'nyatarasyām*
8.4.63 *śaścho 'ṭi*
8.4.65 *jharo jhari savarṇe*

SanskritTagger: A Stochastic Lexical and POS Tagger for Sanskrit

Oliver Hellwig

Institut für Sprachen und Kulturen Südasiens, Freie Universität Berlin, Germany

Abstract. SanskritTagger is a stochastic tagger for unpreprocessed Sanskrit text. The tagger tokenises text and performs part-of-speech tagging using a Markov model. Parameters for these processes are estimated from a manually annotated corpus that currently comprises approximately 1,500,000 words. This article sketches the tagging process, reports the results of tagging a few short passages of Sanskrit text and describes further improvements of the program.

This article describes the design and function of SanskritTagger, a tokeniser and part-of-speech (POS) tagger analysing "natural", i.e., unannotated Sanskrit text, by repeated application of stochastic models. This tagger has been developed during the last few years as part of a larger project for the digitalisation of Sanskrit texts (cmp. [3]) and is still in the state of steady improvement. This article is organised as follows: Section 1 gives a short overview over linguistic problems found in Sanskrit texts that influenced the design of the tagger. Section 2 describes the implementation of the tagger. In section 3, the performance of the tagger is evaluated on short passages of text from different thematic areas. In addition, this section describes possible improvements in future versions.

1 Introduction

Concerning its analytical abilities, the SanskritTagger is located quite low in a hierarchy of taggers. The tagger constructs neither a complete nor a partial syntactical analysis of a Sanskrit text. Instead, it only identifies the most probable lexical resolution for a given group of strings (tokenisation) and their most probable part-of-speech (POS) tags. In comparison with taggers for some European languages, this result might not seem particularly noteworthy. In fact, the limited abilities of this tagger are caused by the difficulties that Sanskrit poses to any tagging process, especially during tokenisation. These problems are not encountered (at least to such a high degree) in the processing of European languages.

On a low phonological level, the euphonic rules called *saṃdhi* are a serious obstacle to an easy tokenisation of Sanskrit text. While these regular phonological transformations can be resolved using finite automata [4] or a simple lookup strategy (see below, 2.2), they introduce a great deal of ambiguity in any analysis of Sanskrit text. Consider, for example, a long string where three points for

G. Huet, A. Kulkarni, and P. Scharf (Eds.): Sanskrit CL 2007/2008, LNAI 5402, pp. 266–277, 2009.

saṃdhi splitting can be identified. Each of these *saṃdhi*s may be resolved in three different ways. Even in this simple example, $3 \cdot 3 \cdot 3 = 27$ new strings are generated by complete resolution at the three splitting points.

The high number of candidate strings that must be checked for validity after *saṃdhi* resolution leads directly to a group of interconnected phenomena that are, in my opinion, the central challenge for any automatic processing of Sanskrit: namely, the extreme morphological and lexical richness of Sanskrit. Even if a moderately sized dictionary is used as in SanskritTagger, there exist about five million distinct inflected nominal and verbal forms that may be found in any text. (English, a language with a large vocabulary, has about one half of this number!) On one hand, in contrast with languages such as German and English, the rich morphology clarifies the functions of words in a phrase and therefore makes POS tagging (and parsing) easier. On the other hand, it is responsible for many analyses that are simply nonsensical (e.g. *āsane* ⇒ *ā* - *sane*, " to - in the gain"). These problems are aggravated by the peculiarities of Sanskrit lexicography. The first important lexical phenomenon is the low text coverage of Sanskrit vocabulary. Compare the following figures for English texts (taken from [7]) and for the Sanskrit corpus I collected during the last years:

Vocabulary size	Text coverage	
	English	Sanskrit
1000	72.0	60.8
2000	79.7	70.3
3000	84.0	75.2
4000	86.8	78.2
5000	88.7	80.3
6000	89.9	81.9

Although the English corpus is certainly better balanced than the Sanskrit corpus, meaning that texts from more diverse sources are included, which should actually lead to a decrease of text coverage, the values for text coverage are clearly higher for English than for Sanskrit. Therefore, when tokenising even a simple Sanskrit text, a tagger must take into account a considerably higher number of lexemes than in other languages. This fact excludes to some extent "easy solutions" such as reduced vocabularies that have proved useful in tagging (technical) texts in other languages.

In addition, Sanskrit possesses a large number of homonyms. A query in the program dictionary that takes into account only homonymous words with the same grammatical category results in the following figures:[1]

[1] This is actually quite a strong constraint, as, for example, nouns of categories "a masc." and "a neutr.", which have the same unchangeable stem, differ only in few forms. Including these pseudo-homonymous words would increase the rate of homonyms up to ten percent of the total vocabulary.

nr. of homonyms	frequency
2	1949
3	112
more than 3	17

Among these homonyms, many high frequency words are found, for instance, *kesara*, m. masc. having the four basic meanings "mane", "(lotus) fibre", "(a plant name, prob.) Mesua ferrea L." and (infrequent) "name of a mountain" (but see LiPur, 1, 72, 7 for a reference).

A further, often neglected, difficulty is the almost complete lack of punctuation marks in Sanskrit texts. Apart from *daṇḍa*s in narrative texts, which often mark the end of a (complex) narrative substructure, Sanskrit texts do not use any kind of reliable punctuation. *daṇḍa*s in metrical texts mark the end of a verse that often, but not regularly, coincides with the end of a syntactic (sub-)structure. Therefore, *daṇḍa*s may be helpful in generating hypotheses about the syntactic structure of a text, but cannot be considered as punctuation marks in a strict sense. This absence of punctuation has a far reaching effect on any tagging or parsing process applied to a Sanskrit text, since all words necessary for a complete analysis may not be contained in the text delimited by these marks. Manual preprocessing of the text (e.g. insertion of a clear punctuation) can counteract this phenomenon, but is certainly contrary to the notion of natural, i.e. unpreprocessed text.

Finally, to understand Sanskrit texts correctly, it is often necessary to supplement a great deal of implicit knowledge. This situation occurs in two closely related areas. First, texts that simulate speech acts, such as dialogues, often make necessary the addition of central parts of speech. This phenomenon known from texts in other languages is a still puzzling, yet intensively studied topic in computational linguistics. Second, scientific texts in Sanskrit such as commentaries or *sūtra*s frequently use a kind of prose that imitates an oral controversy between the proponents of differing opinions. Although this kind of prose is probably derived from real discussions, it uses a highly formalised language (for a description see e.g. [2]). In many cases, "sentences" in this language offer only scant pieces of information that must be inserted in an implicit "knowledge frame" supplied by the reader. Consider, for example, the discussion of *sāpiṇḍya* in the Parāśarasmṛtiṭīkā (on Parāśaradharmasaṃhitā, *Ācārakāṇḍa*, 2, 15; [5], 59). After the author has proposed the standard model of this kind of relation, one that includes three generations projecting into past and future starting from the *yajamāna*, an opponent objects that brother, uncle etc. of the *yajamāna* are not included in this model, and therefore not related to the *yajamāna* by *sāpiṇḍya*. In the following reply of the author, information which is explicitly supplied in the text is printed in bold characters:

maivam
This is **not like this**!
uddeśyadevataikyena kriyaikyasyātra vivakṣitatvāt
Brother, uncle, etc. are included in this model because **the identity of the ritual is expressed by the identity of the gods invoked**.

I am not arguing that these phrases are not well formed. However, their syntax and pragmatics can only be analysed correctly after the pragmatics of the surrounding text has been analysed and "understood" by the computer. The same holds for phenomena such as anaphora resolution. In fact, this is a task that, in my opinion, is far too difficult for any language analysis algorithm currently available.

2 Implementation of the Tagger

To keep data and algorithms clearly separated, language specific information is stored in a database, while the tagging routines are implemented in C++ with heavy use of STL classes. The following section describes these two central components of the tagging software.

2.1 The Database

The first main component of the program, a relational database that can be queried via SQL, stores a dictionary, grammatical information, and a text corpus. The original dictionary was based on the digitized version of Monier-Williams, which was parsed using regular expressions to extract lexemes, meanings, and grammatical categories. These information types were stored in separate tables in the database. During the last few years, the dictionary has been extended in the areas of *Āyurveda* and religious philosophy. It currently contains about 178,000 lexemes (172,000 nouns and 6,000 verbs) with about 185,000 associated grammatical categories and about 325,000 meanings.

An important issue in the processing of strongly inflectional languages such as Sanskrit is the correct recognition of inflected forms. Here, I chose a twofold strategy. Inflected *nominal forms* are not stored in the database, but are recognized on the fly during the tagging process. For this task, all possible endings for any nominal grammatical category are stored in a separate table. During tagging, the last few letters of a given string are compared with these endings. If an ending matches the last letters of the string, the dictionary is searched for the first part of the string in the respective grammatical category. If a matching lexeme could be found in the dictionary, a new candidate is added to the set of possible solutions. Computationally, this approach speeds up the tagging process, because the lookup of the last few letters of a given string in an efficiently organised small set of endings is much less time consuming than a query from a database of over four million inflected forms. Although the difference in duration only amounts to a few milliseconds per operation, the performance loss sums to several seconds for a phrase of moderate size. Figure 1 sketches an example for this approach.

In constrast, inflected *verbal forms* are stored in the database. Currently, the database contains about 440,000 inflected verbal forms including forms derived from prefixed verbs. The decision to store the full verbal forms was not only motivated by the comparatively small number of forms, but also by the frequent

*sanābhy**ām***

ām = acc. sg. of declension type **ā fem.**
dictionary lookup for (*sanābh*, **ā fem.**)
success ⇒ 1st candidate

*sanābh**yām***

yām = loc. sg. of declension type **i adj.**
dictionary lookup for (*sanābh*, **i adj.**)
success ⇒ 2nd candidate

...

Fig. 1. Example for the analysis of nominal forms

irregularities and exceptions in the verbal system of Sanskrit. Of course, it is possible to construct automata generating and accepting correct verbal forms at runtime. However, it seems to me that such an approach requires more effort than the design of a simple algorithm that generates correct verbal forms of the most typical grammatical classes and leaves the rest of the job (including correction of errors and input of rare or special forms) to the user. As in the case of nominal forms, the last few letters of possible verbs are checked before the database is queried.

Apart from the lexical and grammatical information, the database also stores a corpus of analyzed Sanskrit texts, which is of central importance to the tagging process. In short, every separable string of an input text is stored as a separate item in this corpus. After tagging a text, each of its strings is connected with an ordered list of references to grammatically annotated nouns from the dictionary and/or verbal forms. For example, the string *gacchatīśvarātmamāyayā* is analysed and stored as sketched in figure 2. This figure clarifies the first important area of application of this corpus: every string is resolved into lexemes or tokens that are connected to the dictionary. The dictionary points, in turn, to further information about these lexemes as meanings, etc. As these relations may be inverted, the corpus may be used to retrieve Sanskrit words efficiently in large amounts of text. Examples are the retrieval by lexeme ("Show all references in text X for the word Y!"), by meaning ("Show all references of words

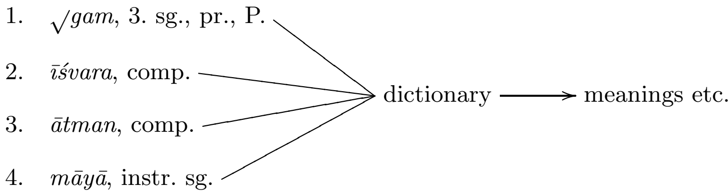


Fig. 2. Analysis and storage of *gacchatīśvarātmamāyayā*

having the meaning X!") or even by semantic concepts. Additionally, the corpus is used to estimate the statistical parameters for the tagging process (see below). At the moment, the corpus contains about 1,560,000 strings which are resolved into 2,190,000 tokens. Each of these tokens is connected with a lexeme, and about 90 percent of them also have a POS annotation. Among other things, the corpus includes the full analysed text of the RĀMĀYAṆA, the first Books of the MAHĀBHĀRATA, some works of *dharma*- (e.g. MANUSMṚTI) and Purāṇa tradition, philosophical literature of Śaivism and many works on *Āyurveda* and *Rasaśāstra* (alchemy).

2.2 The Tagging Algorithm

The tagging software contains two main modules. In the first module, hypotheses about the analysis of a phrase are generated using *saṃdhi* resolution and dictionary lookup. In the case of Sanskrit, the word "phrase" does not mean a complete, self-contained syntactic structure, which may, e.g., be extracted from a text with regular punctuation (see 1). Instead, a phrase $\mathcal{P}$ is a group of strings (words separated by blanks) that is terminated by a (double) *daṇḍa*. Such a group may coincide with a complete syntactic structure. The hypotheses resulting from this first analysis are organised in a frequently complex tree-like or, more accurately, forest-like structure. The purpose of the second module is to find the most probable lexical and morphological path through this structure, given the statistical information extracted from the corpus. This path will constitute the final analysis of the phrase.

A string $\mathcal{S}_i \in \mathcal{P}$ of length L is parsed from left to right. At each position j with $1 \leq j \leq L$, a maximal number of n_{max} letters of the string are searched in a trie $\mathcal{T}$ whose nodes are sorted in binary order. $\mathcal{T}$ stores *saṃdhi* rules of the form $\mathcal{R} = \{s_{SRC}, s_1, s_2, type\}$, where s_{SRC} is the result of the *saṃdhi* between s_1 and s_2, and *type* denotes the area of application of this *saṃdhi* (word, phrase, both). In trie terminology, s_{SRC} constitutes the path that leads to the leaves consisting of the reduced rules $\mathcal{R}' = \{s_1, s_2, type\}$. Multiple leaves may be assigned to a single path, as, for example, $\{\bar{a}, a, both\}$ and $\{a, a, both\}$ to the single letter path $\bar{a}$. n_{max} denotes the length of the longest s_{SRC}, and is pre-calculated at program start.

If an extract s_{ijl} of $\mathcal{S}_i$ at position j matches a path s_{SRC} from the trie, l letters are removed from $\mathcal{S}_i$ beginning at position j. $\mathcal{S}_i$ is split into two new strings $\int_{i1}$ and $\int_{i2}$ at position j and the *saṃdhi* replacements s_1 and s_2 are affixed respectively prefixed to $\int_{i1}$ and $\int_{i2}$. In this way, the new strings $\int_{i1} + s_1 = \mathcal{S}_{i1}$ and $s_2 + \int_{i2} = \mathcal{S}_{i2}$ are created. Figure 3 shows an example for this approach.[2] To keep the *saṃdhi*-rule base simple, the program uses a recursive strategy for *saṃdhi* resolution. For example, a string $\mathcal{S}_{i1} = xxxd$ resulting from the rule $\mathcal{R} = \{dbh, d, bh, both\}$ can be transformed further by application of $\mathcal{R} = \{d, t, -, phrase\}$ into the form *xxxt*. After $\mathcal{S}_{i1}$ and $\mathcal{S}_{i2}$ have been created,

[2] The expression s_{i71} in figure 3 is not a mistake. Because *ch* is treated as a single phoneme, it is replaced with a single letter in the internal representation.

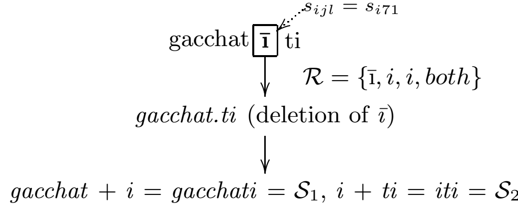


Fig. 3. *saṃdhi* resolution for the string *gacchatīti*

the *saṃdhi* routine is recursively called for these new strings, which are treated in the same way as $\mathcal{S}_i$. If the running index j reaches the end of a string ($j = L$), the program checks if $\mathcal{S}_i$ is a valid Sanskrit form. The procedures for this check and the necessary structures of the database were shortly described in section 2.1. If $\mathcal{S}_i$ is a valid Sanskrit form, the grammatical and lexical analysis of $\mathcal{S}_i$ are inserted into the analysis of $\mathcal{P}$.

During *saṃdhi* resolution and dictionary lookup, each $\mathcal{S}_i \in \mathcal{P}$ may have been resolved into m different subsets $\{\mathcal{S}_{i11}, \mathcal{S}_{i12}, \ldots, \mathcal{S}_{i1n_1}\}$, ..., $\{\mathcal{S}_{im1}, \ldots, \mathcal{S}_{imn_m}\}$ due to different *saṃdhi* resolution (SR). Additionally, each of the substrings $\mathcal{S}_{ijk}$ has at least one grammatical and lexical analysis A_{ijkl} attached to it. Here, j is the number of the current SR, k is the position of the substring in SR_j, and l is the index of the analysis for substring $\mathcal{S}_{ijk}$. Take, for example, the string $\mathcal{S}_1 =$ *devadattakṛtakriyā*. (Parts of) two possible solutions (SR_1 and SR_2) are shown in Figures 4 and 5. As indicated by the lines connecting the partial solutions A_{ijkl}, the second step of the tagging process consists in finding the most probable path that connects all $\mathcal{S}_i \in \mathcal{P}$. This step is again divided into two substeps. In the first substep, the most probable lexical path is searched with the help of a Markov model (MM). This path is fixed as the tokenisation of $\mathcal{P}$. In the second substep, the most probable syntactical analysis of this path is searched for with another MM.

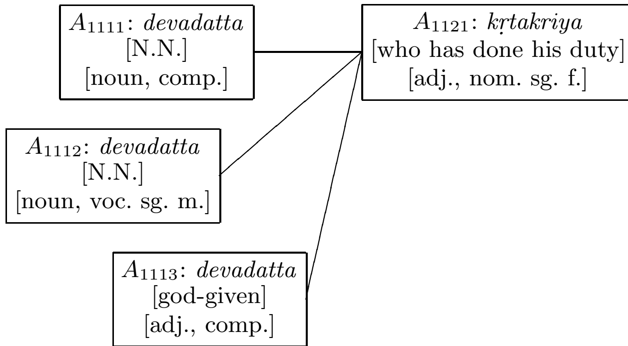


Fig. 4. A possible analysis of the string *devadattakṛtakriyā*

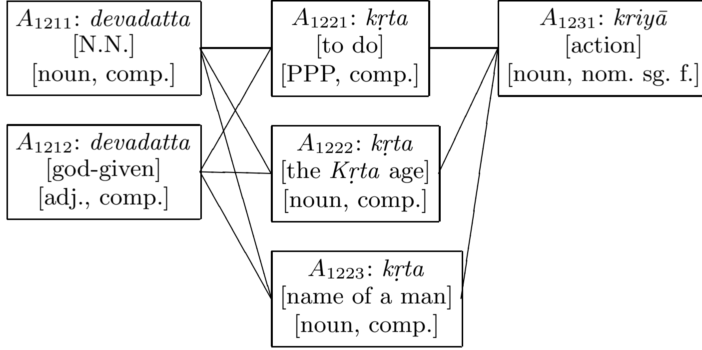


Fig. 5. Another possible analysis of the string *devadattakṛtakriyā*

The first substep, i.e. the *tokenisation*, can be modelled using a discrete, first-order Markov chain (see e.g. [6] for a readable introduction). The MM is based on the concept of conditional probability, which is the probability of event B given the occurrence of another event A: $P(B \mid A) = \frac{P(A \cap B)}{P(A)}$. If $x_{[a,b]}$ denotes an ordered sequence of lexemes with decreasing indices i ($a \geq i \geq b$), the probability that phrase $\mathcal{P}$ of length L is tokenised into lexemes $x_1, x_2, \ldots x_L$ is given by

$$\begin{aligned} \underbrace{p(x)}_{P(A \cap B)} &= \underbrace{p(x_L \mid x_{[L-1,1]})}_{P(B|A)} \cdot \underbrace{p(x_{[L-1,1]})}_{P(A)} \\ &= p(x_L \mid x_{[L-1,1]}) \cdot p(x_{L-1} \mid x_{[L-2,1]}) \\ &\quad \cdot p(x_{[L-2,1]}) \quad \text{etc.} \end{aligned} \tag{1}$$

Under the assumption that the probability of each lexeme depends only on its direct predecessor (first-order model), the formula is simplified to

$$\begin{aligned} p(x) &= p(x_L \mid x_{L-1}) \cdot p(x_{L-1} \mid x_{L-2}) \cdot \ldots \cdot p(x_1) \\ &= p(x_1) \Pi_{i=2}^{L} p(x_i \mid x_{i-1}) \end{aligned} \tag{2}$$

For reasons of floating point accuracy, equation 2 is transformed into

$$p(x) = \log p(x_1) + \sum_{i=2}^{L} \log p(x_i \mid x_{i-1}) \tag{3}$$

with $\log(a \cdot b) = \log a + \log b$. To find the most probable path, a modified form of the well known *Viterbi algorithm* is used. This algorithm was actually designed for HMMs, but can be applied to the given problem due to its similar structure. Before applying this algorithm to the data, it should be taken into consideration that any string $\mathcal{S}_i$ may have been resolved in subsets S_{ix} and S_{iy} with different sizes $|S_{ix}| \neq |S_{iy}|$. Therefore, the algorithm cannot be applied naively to the hypotheses generated in the first step. Instead, for each phrase $\mathcal{P}$ that contains

N strings $\mathcal{S}_1, \ldots, \mathcal{S}_N$ a vector of length N is allocated storing the currently checked index j of its *saṃdhi* resolutions for every string $\mathcal{S}_i$. A vector starting with $\boxed{1}\boxed{2}\boxed{1}$ ... means that the first resolution of $\mathcal{S}_1$, the second resolution of $\mathcal{S}_2$, and the first resolution of $\mathcal{S}_3$ are checked. Such a combination of the analyses of successive strings will be called *path subset* (PS). (Note that the diagrams which show possible resolutions of *devadattakṛtakriyā* each constitute one path subset!) If n_i denotes the number of SRs for $\mathcal{S}_i$, there exist $\Pi_{i=1}^{N} n_i$ different PSs. Next, among all PSs, the PS having the best path regarding lexical probability is searched with a modified version of the Viterbi algorithm. Some PSs consist of shorter paths because the strings in these combinations were split at fewer *saṃdhi* points. To treat all PS in the same way, they are filled up with "dummy probabilities" that are calculated as the mean of the probabilities constituting the best path $\pi_{opt\,u} \in PS_u$. If l_{max} is the length of the longest PS, l_u is the length of the current PS_u, and $\bar{p}(\pi_{opt\,u})$ is the average transition probability between all elements constituting the optimal path in PS_u, the value $\sum_{r=l_{max}-l_u}^{l_{max}} \log \bar{p}(\pi_{opt\,u})$ is added to the probability of $\pi_{opt\,u}$.

The most probable path found using this algorithm is considered the tokenisation of $\mathcal{P}$, which is annotated morphologically in the next step. The POS tagset used for this annotation contains the 136 items shown in table 1. To understand the relation between this tagset and the actual morphological analysis of a word, consider the string *gacchati*. After the program has fixed the lexeme *gam* ("to go") as its most probable lexical analysis, there are three possible morphological resolutions: "he/she/it goes" (3rd, sg., pres., P.) and "in the going ..." (loc. sg., masc./neutr., part. pres., P.). The solution "he ... goes" is mapped to the POS tag ["present tenses", 3rd, sg.] while the nominal solutions are mapped to ["present participles", loc., sg., masc.] and ["present participles", loc., sg., neutr.], respectively. Note that a good deal of information is lost during this mapping process since, for example, no distinction is made between different present tenses. Instead, forms such as *gacchati* and *gacchatu* are considered syntactically equivalent and are, therefore, mapped to the same POS tag. This loss of information reflects the decision between the granularity of the tagset and the amount of text from which the probabilities of the tags can be estimated – the more text is available, the finer a granularity can be chosen. The optimal morphological path is again searched for using a Markov model and a variant of the Viterbi algorithm. This path is presented to the user as the most probable resolution of a phrase. It may be either accepted and stored in the database or (manually) replaced with a better solution.

3 Performance and Improvements

This section gives the results of tagging some short passages of text. The results are in no way representative. They are only meant to demonstrate the performance of the algorithm on different types of Sanskrit text. Three types of errors

Table 1. Tagset used for POS tagging of Sanskrit text

Verbal forms		
Finite verbal forms		
present tenses (incl. imperative and opt.)	×9: person, number	9
past tenses	×9: person, number	9
future tenses	×9: person, number	9
other tenses	×9: person, number	9
Infinite verbal forms		
absolutive		1
infinitive		1
past participle, gerund	×24: case, nr., gender	24
present participles	×24: case, nr., gender	24
other participles	×24: case, nr., gender	24
Nominal forms		
indeclinable		1
nouns in composite words		1
nouns, adjectives	×24: case, nr., gender	24
		136

are distinguished. A saṃdhi *error* (e_S) occurs if a string is split at incorrect splitting points. This error invalidates the results for the whole string. A *lexical error* (e_L) indicates that a string was split at correct *saṃdhi* points, but that a wrong lexeme was activated during tokenisation. A *POS error* (e_{POS}) occurs if a wrong POS tag was assigned to a correct token. In addition, the value r_{SL} gives the ratio of strings to lexemes, i.e. $r_{SL} = \frac{\text{nr. of strings}}{\text{nr. of lexemes}}$. A high value of r_{SL} indicates that the passage contains few composites. The following five passages were analysed:

1. Liṅgapurāṇa, 2, 20, 1-10: A *Purāṇic* text treating a Śivaite topic. Easy verses.
2. Viṣṇusmṛti, 63, 35-50: An example of the scientific style. Many supplements are needed to get the full meaning of the passage.
3. Mūlamadhyamakārikā of Nāgārjuna, 12, 1-10: Easy Buddhist prose (from a linguistic point of view).
4. Gītagovinda of , 1.2-5: Poetry with many unusual words.
5. Kāmasūtra, 2, 1, 1-12: Scientific prose.

The results, which are displayed in Table 2, support the assumptions about the problems that are encountered in tagging natural Sanskrit text (see section 1). The tagger performs best on texts that are written in an easy style and come from "well known" areas of knowledge (1, 3). In contrast, a difficult vocabulary (5) and demanding syntactical structures (4) introduce a great deal of *saṃdhi* (4, 5) and POS (5) errors. The comparatively high number of POS errors in 3

Table 2. Error rates of the tagger in five short passages – Abbreviations: **nr.**: number of the passage, **P**: number of phrases, **S**: number of strings, **L**: number of lexemes

nr.	P	S	L	r_{SL}	e_S	e_L	e_{POS}	corr. phrases
1	22	89	139	0.64	4	6	2	10
2	17	48	88	0.55	2	4	6	5
3	20	135	157	0.86	0	2	9	13
4	8	49	86	0.57	8	4	1	0
5	24	123	167	0.74	8	5	15	10

is primarily caused by confusion between nom. and acc. sg. neuter and could certainly be reduced by training the tagger with only a few similar texts.

At the moment, three main areas for improving this tagger can be discerned. First, probability values of rare lexemes and infrequent POS combinations must be estimated more reliably. Applying the classical forward-backward-re-estimation actually leads to degradation of the probability values (cmp. [1, 3]). Although many other methods, such as smoothing probabilities or the use of neural networks, were proposed, Sanskrit offers a (partial) solution of this problem that is based on its lexicography. Sanskrit possesses not only a high number of homonymous, but also of synonymous words. Many of these words are already integrated in a semantic network (based on the `OpenCyc` ontology) that is part of the program database. To estimate probabilities, groups of words are identified that designate the same sememe with a high degree of probability. For instance, the group "horse" is constituted by {*aśva*, *turaga*, *turaṃga*, *turaṃgama*, *vājin*, *haya*}, but not *hari*, which means "Viṣṇu" in most cases. If one of the words that is included in the group "horse" is met in an unknown context (either lexical or POS/morphological), the respective probabilities can be estimated from the values given for other members of the group.

Second, integration of rules can certainly improve analysis. Due to the lack of punctuation marks (see 1), these rules should not describe well-formed and complete phrases, but only check the coherence of few members of a phrase $\mathcal{P}$, i.e. a syntactic substructure delimited by *daṇḍa*s (chunk parsing). Some preliminary tests using rules that reject paths during POS tagging on the basis of simple syntactic criteria turned out to be successful.

Third, the strict separation of tokenisation and POS tagging is a constant source of errors. Consider, for example, the simple sentence *brahmā varaṃ te dāsyati* ("Brahmā will give you a boon."). Although the POS sequence [dat. sg.] - [3. sg. fut.] is well established and would always be preferred to the incongruent [nom. pl.] - [3. sg. fut.], *te* is interpreted as the nom. pl. masc. of the pronoun *tad* due to its enormous frequency during tokenisation. Since the content of the best lexical path can not be changed during the following POS analysis, the correct analysis for *te* (dat. sg. of *tvad*) will never be activated. Possible workarounds for this problem are a more flexible POS analysis that takes into account the first five lexical resolutions or a combination of tokenisation and POS tagging in one procedure.

References

1. Abney, S.: Part-of-speech tagging and partial parsing. In: Church, K., Young, S., Bloothooft, G. (eds.) Corpus-Based Methods in Language and Speech. Kluwer Academic Publishers, Dordrecht (1996)
2. Hartmann, P.: Nominale Ausdrucksformen im wissenschaftlichen Sanskrit. Carl Winter Universitätsverlag, Heidelberg (1955)
3. Hellwig, O.: Sanskrit und Computer. PhD thesis, Freie Universität Berlin (2002)
4. Huet, G.: Shallow syntax analysis in Sanskrit guided by semantic nets constraints. In: Proceedings of the 2006 International Workshop on Research Issues in Digital Libraries, Kolkata, West Bengal, India. ACM, New York (2007)
5. Vāmanaśarmā, I.V. (ed.): Parāśaradharmasaṃhitā. Ācārakāṇḍam (1893)
6. Rabiner, L.R.: A tutorial on Hidden Markov Models and selected applications in speech recognition. Proceedings of the IEEE 77, 257–286 (1989)
7. Waring, R., Nation, P.: Schmitt, N., McCarthy, M. (eds.) Vocabulary: Description, Acquisition and Pedagogy, pp. 6–19. Cambridge University Press, Cambridge (1997)

A Glimpse into the *Apadam*-Constraint in the Tradition of Sanskrit Grammar

Prasad P. Joshi

Fergusson College, Pune 411 004, India
joshiprasadp@yahoo.co.in

Abstract. The tradition of Sanskrit Grammar has put a constraint: *apadaṁ na prayuñjīta* (non-inflected form should not be used in the sentence). The present paper is an attempt to discuss its salient features, while showing its assumption in Pāṇini's model and comparing it with the contemporary theory. A form which is suffixed with *sup* and *tiṅ* is termed as *pada* by Pāṇini. Every lexical item is tagged as *dhātu* or *prātipadika* in his system and subsequently affixed with the inflections and thus is furnished as a *pada,* fit to enter into a sentence. It seems that the *apadam*-constraint is an effective tool to eliminate the ill formed sequences from the well formed sentences thus contributing to the generative theory. The case filter principle of the modern universal case theory, with some difference, bears striking similarities with the present constraint. It reads: every overt NP must be assigned abstract case. However, although very effective, it is concerned with NPs alone. The *apadam*-constraint, although language specific, has a wider application being concerned with both NP and VP.

Keywords: *Apada*, non-inflected form, *pada* in the Pāṇinian model, Case- Filter Principle, abstractness, universal application.

1 Introduction

The tradition of Sanskrit Grammar reads a constraint *apadaṁ na prayuñjīta* which means that a non-inflected form should not to be used (in a sentence). Positively it means only inflected forms are to be employed in a sentence. Here the word *apada* is regarded as a noun; alternatively it is possible to regard it as an adjective (*avidyamānaṁ padaṁ yasmin tad apadam*), meaning 'that which is devoid of a *pada*'. In this case it qualifies the substantive sentence (*vākya).* Thus, the constraint can be interpreted in two ways, viz. (i) one should not use a non-inflected form (in a sentence) (ii) one should not employ a sentence devoid of inflected forms. The constraint no doubt brings out the notion that a sentence includes only inflected forms. Non–inflected forms cannot enter into a sentence. The present paper is an attempt to discuss the salient features of this constraint, while showing its assumption in Pāṇini's model and comparing it with the contemporary theory.

2 Occurrence of the Present Constraint

The present constraint is handed over from generation to generation in the oral tradition of Sanskrit learning. Eminent scholars of Sanskrit Grammar trained in the

G. Huet, A. Kulkarni, and P. Scharf (Eds.): Sanskrit CL 2007/2008, LNAI 5402, pp. 278–286, 2009.

traditional ways like (late) Pandit V.B. Bhagwat (Pune) and Dr. S.D. Joshi (Pune), who combine both traditional as well as modern idioms, to name a few, are proofs to this. The constraint has been quoted by Nāgeśa in his *Paramalaghumañjūṣā* as a statement in the *Mahābhāṣya*. In the section of *Nāmārthavicāra*, he says: *nanu apadaṁ na prayuñjīta iti bhāṣyād asādhv idam iti cet na.*[1]

However, the origin of the present constraint is not traced in the *Mahābhāṣya*, nor in any other written treatises. Nevertheless, it is possible to trace some quotations in the Sanskrit literature which implicitly refer to the present constraint; e.g. Sureśvara in his *Naiṣkarmyasiddhi* reads: *na samasty apadaṁ vākyaṁ yat syāt jñānavidhāyakam.*[2] Which means there cannot be a sentence which is devoid of a *pada* and yet expressive of knowledge.

3 What Is an *Apada*

To ascertain the exact import of the constraint and also to decide whether it is a semantic or syntactic one, it is essential, at the outset, to consider the term *apada*. In the plain sense it means non-*pada*. The term *pada* is variously treated in Nyāya, Vyākaraṇa and also in some Poetic treaties. In the tradition of Nyāya it normally refers to a meaningful unit. Annambhaṭṭa in his *Tarkasaṅgraha* defines *pada* as *śaktaṁ padam,*[3] which means *pada* is a (linguistic) unit which is competent (of expression). Thus, the sequence of phonemes which does not convey any meaning is an *apada*. The position of Nyāya is purely semantic. If viewed from the Naiyāyika's definition of the term *pada*, the constraint means: 'a meaningless unit should not be used', becoming a purely semantic one.

In Vyākaraṇa a form which is suffixed with *sup* and *tiṅ* is termed as *pada* by Pāṇini (*suptiṅantaṁ padam,* P.1.4.14).[4] In his system *sup* is the term for nominal suffixes taught in the rule *svaujasamauṭ*... etc. (P.4.1.2) and *tiṅ* refers to the verbal endings taught in that of his *tiptasjhi*... etc. (P.3.4.78). In Vyākaraṇa, since *pada* means an inflected form either nominal or verbal, *apada* means a non-inflected form, i.e. a base or stem. Pāṇini's concept of pada is morphological. The present constraint, if viewed from the Pāṇinian terminology of pada, becomes purely a grammatical one.

The *apadam*-constraint can be a semantic or syntactic one as shown above. As a semantic constraint it will be very basic and general in nature and will hardly require any explicit statement. It will not mark any linguistic peculiarity of Sanskrit. On the other hand, as a syntactic constraint it plays a particular role for Sanskrit syntax. It will mark the indispensability of inflections in the Sanskrit morphology.

---

1 Bhagwat, V.B. (ed.): *Paramalaghumañjūṣā* Part 1. Parāmarśa Prakāśanamālā 4, Pune University, pp.100, Pune (1984).

2 Jacob, G.A. (ed.): *Naiṣkarmyasiddhi*. BSPS 38, pp. 14, Bombay (1985).

3 Athalye, Y. V. (ed.): *Tarkasaṅgraha*. BSS 50, pp. 50, Bombay (1930).

4 Pāṇini extends the term *pada* further in three aphorisms to refer to a special kind of nominal stem, which is to have specific morphophonemic changes. But this definition is not relevant for the present constraint.

4 The Present Constraint and Pāṇini

Pāṇini's *Aṣṭādhyāyī* (c. 400 BC.) basically deals with the derivational morphology. It is a generative grammar that constructs speech forms from basic elements. In the Pāṇinian system the noun *rājapuruṣa* is derived as *rājan+as +puruṣa+as;* or the verb *bhavati* as *bhū+a+ti*. This system deals with verbal and nominal morphology. Pāṇini's concept of pada is already referred to above. It will be appropriate at this point to see how he furnishes basic elements as padas. The basic elements are three, viz. verbal bases, nominal bases and feminine bases.

4.1.1 Verbal Bases

In the Pāṇinian system a verbal root is termed as *dhātu.* His rule *bhūvādayo dhātavaḥ* (P.1.3.1.) assigns the term *dhātu* to the primary roots *bhū-* etc. and another rule *sanādyantāḥ dhātavaḥ* (P.3.1.32) assigns the same to the secondary roots furnished with suffixes -*san* etc. This covers the following formations: 1) a causal base ending in *–ṇic*, e.g. *bhāvay-* (<*bhū-* 'to cause to be'); 2) a desiderative base ending in -*san*, e.g. *bubhūṣa-* (<*bhū-* 'to desire to be') (P. 3.1.5); 3) a frequentative base ending in -*yaṅ* and -*yaṅluk*, e.g. *bobhūya-* (<*bhū-* 'to be intensively'); 4) and various denominative bases ending in *-kyac, -kāmyac, -kyaṅ, -kyaṣ*, etc. (P. 3.1.8 etc.), respective examples: *putrīya-* ('to desire a son for oneself'), *putrakāmya-* ('to desire a son for oneself'), *apsarāya-* ('to behave like a nymph'), *lohitāya-* ('to become red').

4.1.2 Nominal Bases

In the system of Pāṇini a nominal base is termed as *prātipadika*. His rule *arthavad adhātur apratyayaḥ prātipadikam* (P.1.2.45) assigns this designation to a meaningful unit which is not a verbal base, not an affix, nor one ending in an affix, e.g. *go* ('cow'), *aśva* ('horse') etc. The next rule, *kr̥ttaddhitasamāsāś ca* (P.1.2.46), extends the term further to apply to the primary derivatives ending in *kr̥t* suffixes, secondary derivatives ending in *taddhita* suffixes and to compounds (*samāsa*). Respective examples are as follows: *kartr̥* ('maker') derived from *kr̥-* with the suffix *–tr̥c* (P. 3.1.133.); *āśvapata* ('descendent of *aśvapati'*) derived from the nominal stem *aśvapati* with the suffix *aṇ* (P. 4.1.84); *rājapuruṣa* (king's man) a compound of *rājan* + *puruṣa* (P. 2.2.8).

4.1.3 Feminine Bases

Nominal and verbal terminations (*sup* and *tiṅ*) occur after a small class of items other than dhātus and prātipadikas, namely, after speech forms terminating in feminine suffixes -*ī* and -*ā*. Such speech forms do not receive the designation *prātipadika* in accordance with P. 1.2.45-46 because they terminate in an affix other than one termed *kr̥t* or *taddhita* and they are not compounds (*samāsa*). These feminine bases are referred to as simply *ṅyanta* (ending in *ī*) and *ābanta* (ending in *ā*) bases.

4.2 Basic Elements Developed as Padas in the Pāṇinian System

As a next step all the three bases receive inflectional suffixes. Inflections are of two kinds, viz. sup and tiṅ. They are technically termed as *vibhakti* (P.1.4.104). sup-vibhaktis are the nominal suffixes *svaujasamauṭ*… etc. coming after prātipadikas and feminine *ṅyanta*, *ābanta* bases (P. 4.1.1). Tiṅ–vibhaktis are the verbal endings *tiptas-jhi*…etc. coming after dhātus. As a second step every lexical item becomes finished with either sup or tiṅ-vibhaktis in the Pāṇinian system. At this stage they are termed as *pada*-s (inflected forms) according to P. 1.4.14.

4.3 *Avyaya* as Pada in the Pāṇinian System

Here it is important to note that even the so–called indeclinable of modern syntax is termed as *vibhaktyanta pada* in Pāṇini. Words like *ucchaiḥ* (‘high’), *nīcaiḥ* (‘low’) which are classed as indeclinables according to the modern syntax are looked upon as prātipadikas as a first step in the Pāṇinian system (P.1.2.45). According to P.2.3.46 they are appended with the nominative case suffix to convey the mere nominal stem meaning.[5] This nominative case is subsequently zeroed by *luk* elision according to P.2.4.82, and the words become covertly marked for the nominative sup-vibhakti. By his rule *pratyayalope pratyayalakṣaṇam* (P.1.1.62), they are treated as *vibhaktyanta*-s and hence receive the status of a pada. The so-called indeclinable of modern syntax is technically termed as *avyaya* by Pāṇini.

Here it is essential to mention that Pāṇini does not indulge himself in giving a functional definition of *avyaya* (like *yan na vyeti tad avyayam*, etc.). Instead, Pāṇini has given an enumerative definition of *avyaya*.[6] His five rules beginning with *sva-rādinipātam avyayam* P. 1.1.37 enumerate the words which will have the designation of *avyaya*. They include nipātas (particles) like *ca* (‘and’), upasargas (prepositions) like *pra*, case adverbs like *tatra* (‘there’), infinitives like *kartum* (‘to do’), gerunds like *kr̥tvā* (‘having done’), and adverbial compounds like *adhihari* (‘on the Lord Hari’). Pāṇini recognizes them as subclasses of the broadest *avyaya*-class.

4.4 Pada as a Single Class in the Pāṇinian System

The fact that Pāṇini regards avyayas to be covertly marked for vibhakti is suggestive of his attempt to squeeze all the parts of speech into a single category of pada. This pada is only twofold *tiṅanta* (finite verb) and *subanta* (inflected noun). *tiṅanta*-s include *bhvādi tiṅanta* (primary) and *sanādi tiṅanta* (secondary). *subanta*-s include

---

5 P.2.3.46 includes the expression *prātipadikārtha*. The point in this kind of process is that the words like *ucchaiḥ* (‘high’), *nīcaiḥ* (‘low’) which do not possess a gender or number of their own, neither being fit to be marked for any specific case meaning, are appended nominative in the mere sense of prātipadika and assume the status of a pada.

6 His definition of *sarvanāman* is also of an enumerative kind and not a functional one, cf. P. 1.1.27.

prātipadika-subanta, *kṛt-subanta* (primary derivatives), *taddhita-subanta* (secondary derivatives), *samāsa-subanta* (compounds) and *ṅyanta-*, *ābanta-subanta* (feminine derivatives).

Every word, according to Pāṇini, has to have a pada-status. His model is designed in such a way that out of it every basic element must be derived as pada. It seems that Pāṇini himself must have presupposed the principle *apadaṁ na prayuñjīta*, or some commentator on Pāṇini has postulated it in order to teach the morphological process of Pāṇini. Be it as it is, the constraint *apadaṁ na prayuñjīta* underlines the technique of word derivation used by Pāṇini in his *Aṣṭādhyāyī.*

5 The Present Constraint and Sanskrit Language

The constraint *apadaṁ na prayuñjīta* reflects upon the nature of Sanskrit which is highly inflectional; e.g. in a Sanskrit sentence: *Devadattaḥ kaṭaṁ karoti* ('Devadatta makes a mat'); *Devadattaḥ* is the pada being finished with the case inflection *-su*. This single suffix stands for two units of meaning, viz. agent and singular, i.e. it marks two categories of case and number. Similarly *karoti* is the pada finished with the verbal inflection *-tip,* which marks the agent, number, tense and voice simultaneously. Thus in Sanskrit, the vibhakti (inflectional part) is always more expressive. Besides marking the syntactic categories, the vibhaktis, especially the sup-vibhaktis, carry another important function of revealing the *kāraka*-roles (if it is not expressed by a tiṅ suffix) and other structural relations.

5.1 *Kāraka*-Roles Exemplified

Kāraka-s are the specific syntactic-semantic relations that the NPs have with the predicate VP. The tradition of Sanskrit grammar specifies six kāraka-roles. They are *kartṛ* ('agent of the action'), *karman* ('object or goal of the action'), *karaṇa* ('instrument of the action'), *sampradāna* ('beneficiary of the action'), *apādāna* ('point of separation of the action'), and *adhikaraṇa* ('substratum of the action'). For instance, in the sentence *Devadattaḥ kaṭaṁ karoti* ('Devadatta makes a mat'), *Devadattaḥ* is related with the verb *karoti* by being an *abhihita kartṛ* ('expressed agent') of the activity, *kaṭam* is related with the verb *karoti* by being a karman (object) to be obtained by the action. Thus *Devadatta* and *kaṭa* play the kāraka-roles of kartṛ and karman respectively. These roles are to be revealed in a sentence. The sup-suffix is the device to reveal them. The twenty-one sup-suffixes are divided in triplets and respectively termed as *prathamā*, *dvitīyā*, *tṛtīyā*, *caturthī*, *pañcamī*, *ṣaṣṭhī*, and *saptamī.* Each triplet stands for some specific kāraka-relation in the following way:

dvitīyā denotes karman
tṛtīyā denotes karaṇa,
caturthī denotes sampradāna,
pañcamī denotes apādāna, and

ṣaṣṭhī denotes either kartṛ or karman, P.2.3.65,
saptamī denotes adhikaraṇa.

As a first step, the semantic object denoted by a noun is assigned a kāraka-term. Then to reveal that role an appropriate vibhakti is added. This assignment of vibhakti is determined by the kāraka-role and is known as *kāraka-vibhakti* in the tradition.

5.2 Other Relations Exemplified

The tradition of grammar recognizes another type of vibhakti assignment—the *upapada-vibhakti*—wherein a certain word assigns vibhakti to the nominal stem (*prātipadika*), or feminine-affix ending speech form (P.4.1.1), which is structurally related to it, e.g. *antarā tvāṁ māṁ ca kamaṇḍaluḥ* ('the bowl is between thee and me'). Here the preposition *antarā* ('between') governs the *dvitīyā* of the NPs structurally related to it. In the upapada-vibhakti there is a structural expectancy. Government is the only principle there. Similarly the *ṣaṣṭhī* vibhakti also stands for relations like possessor and possessed, whole and part etc; e.g. *rājñaḥ puruṣaḥ* ('king's man'). In this way there are two types of vibhakti-assignment in Sanskrit, viz. kāraka and akāraka (including *ṣaṣṭhī* and *upapada*). The former reveals the kāraka-roles while the latter reveals structural relations like noun-noun relations among constituents.[7]

6 The Present Constraint and Structure

The very nucleus notion of a structure is that its constituents are related to each other. A sentence is a structure made up of words and phrases that are related to each other. Without their mutual relations it will otherwise be a string of words; it would not make a sentence. It cannot convey any sense either. It is seen here that vibhakti is helpful to reveal this syntactic relation and also to mark various categories of case, number, tense and voice. This notion was realized in the early tradition of Sanskrit grammar. The *apadam*-constraint is the systematic outcome of this realization.

7 Applicability of the Present Constraint

It is a well known fact that the earlier Vedic language (c.1500 BC - 800 BC) is much more archaic in nature, and many of its features are difficult to account for through the Pāṇinian system. In that language very often some forms are available without case terminations, e.g. the locative singular *vyoman* (instead of *vyómni* or *vyómani*), *áśman* (instead of *áśmani*).[8] This type of expression is more frequent in the *ṛgveda*. Naturally the constraint is not obeyed here. Pāṇini regards the *luk*-elision of nominal suffixes in case of such Vedic forms (P.7.1.39). They receive the designation pada by his rule *pratyayalope pratyayalakṣaṇam* (P.1.1.62).

---

7 It is possible to regard the sup-vibhakti as fourfold, viz. *kāraka*, *upapada*, *śe˘a*, and *prātipadikārtha*.
8 Macdonell, A.A.: *Vedic Grammar.* Para.325, Karl J. Trunbar, Strassburg (1910).

In the case of modern Sanskrit poetry also it is possible that the constraint is ignored especially in the case of many foreign words. The present constraint applies much more to the grammatically regular or a standard language.

8 Generative Aspect of the Present Constraint

It seems that the *apadam*-constraint together with the application of Pāṇini's derivative rules can be an effective tool to eliminate the ill-formed sequences from the well-formed sentences thus contributing to the generative theory. In Pāṇini, every lexical item will be tagged and then accordingly affixed with either of the two vibhaktis and thus will be furnished as a pada, fit to enter into a sentence. The constraint will work here. For example it will reject the sequence *Devadatta kaṭa kr̥* ('Devadatta mat make') as not being a sentence since it contains all apadas (bases).While it accepts *Devadattaḥ kaṭaṁ karoti* ('Devadatta makes a mat') as a well-formed sentence since it contains all the padas (i.e. inflected forms).

9 The *Apadam*-Constraint and Modern Universal Theory

It will be interesting to see common speculations between the ancient Indian theory and the contemporary one. The case filter principle of modern universal theory, with some differences, bears striking similarities with the present constraint. The case filter principle reads: "Every overt NP must be assigned abstract case".[9] In plain words it means that in a sentence the NP must be marked with some case that is assigned to it; one cannot employ any non-case-assigned NP in the sentence. This principle is called as *filter* because it filters out the sequences wherein the NP is employed without any case assigned to it.

9.1 Theta Role

Yet another important principle related to universal case theory is that of theta role. It demands attention while dealing with the case filter condition. The universal theory claims that every predicate has argument structure, it is specified for a number of arguments it requires; e.g. in the English sentence 'John killed Bill', *kill* is the predicate; it requires at least two arguments. The NPs *John* and *Bill* are its arguments, so it is a two-way predicate. The arguments are the participants minimally involved in the activity or state expressed by the predicates. Thus they are related with the VP. These relations are looked upon as theta roles. Liliane Haegemen enumerates nine theta roles as agent, patient etc.[10] In the above quoted example *John* has the theta role as agent and *Bill* as that of patient. The theta criterion of the universal theory says that every argument must be assigned a theta role. Case is the device to mark the theta role

---

[9] Haegemen, Liliane: *Introduction to Government and Binding Theory.* pp. 167, Blackwell Publisher Ltd, Oxford (1991).
[10] Ibid. pp. 49.

of the NP. Hence the case is indispensable. Here Liliane Haegemen tries to explain the indispensability of a case in a fine metaphor of the play. She says:

> "The argument NPs must be made visible by means of case in the way that the characters playing a part in a performance must be made recognizable by their outward appearance. If all actors looked identical we would not be able to tell who is playing which part. NPs are licensed by virtue of their case properties".[11]

Here the case assignment is via theta role, and it is called as inherent case assignment. It is distinct from the structural case-assignment wherein an NP governs the case of another NP which is structurally related to it. It is blind to any theta role, e.g. in English the preposition *towards* governs the accusative, as in the sentence: *John went towards him.* This kind of case assignment is referred to as a structural case assignment. The case becomes necessary to mark the theta roles and also the structural relations. The case filter condition is the outcome of this necessity.

10 Comparing the Traditional and Modern Approaches

Certain points of similarity between the *apadam*-constraint of traditional Sanskrit grammarians and the case filter principle of the universal theory have clearly emerged from what has preceded. They hardly require any mention. However the major area where they differ is the notion of abstractness. The case filter condition requires an NP to be assigned an abstract case. It operates on an abstract level. Case is taken to be an abstract property; its morphological realization can vary from language to language. In an inflectionally rich language like Sanskrit, there will be a full-fledged morphological realization of the case. In an analytic language like English, it will be less. Thus the abstraction makes the case filter principle applicable universally. On the other hand, the traditional *apadam*-constraint refers to specific terminations termed *sup* and *tiṅ* in accordance with Pāṇini's abbreviatory technique known as *pratyāhāra*. It becomes specific to Sanskrit, and there is hardly any scope to apply it to other languages in the literal sense. The case filter principle takes into consideration the NPs. It has nothing to say about VPs. Hence it cannot account for the total grammaticality of the sentence. The *apadam*-constraint applies to both NPs and VPs. Strictly speaking all the modern parts of speech are included in these two. Thus *apadam*-costraint becomes a complete principle to check the grammaticality of a sentence.

11 Conclusions

Apadaṁ na prayuñjīta is a syntactic constraint handed over by the oral tradition of ancient Indian grammarians. It is based on the Pāṇinian notion of pada. The constraint is useful to check the grammaticality of a sentence. Actually this is the area where the interests of contemporary linguistics and ancient Indian grammar intersect each other.

---

[11] Ibid. pp.189.

The case filter principle of the modern universal theory runs on almost similar lines. However, although very effective, it is concerned with NPs alone. The *apadam*-constraint, although language specific, has a wider application being concerned with both NPs and VPs.[12] An attempt can be made to bring out a common product, obtaining the best of both worlds, which would surely lead to some positive developments, not merely theoretical but practical also.

Acknowledgements. I am indeed grateful to Dr. S.D. Joshi (Pune) for a variety of information that he has given me voluntarily during the preparation of this paper. I am also thankful to Dr. H.C. Patyal (Pune) for carefully going through the paper and adding his valuable suggestions to it.

References

1. Böhtlingk, O.: Pāṇini's Grammatik. George Olms Verlag, New York (1887)
2. Haegemen, L.: Introduction to Government and Binding Theory. Blackwell Publisher Ltd., Oxford (1991)
3. Joshi, S.D., Roodbergen, J.A.F.: Patañjali's Mahābhāṣya Prātipadikāhnika. CASS-14, Pune (1981)
4. Joshi, S.D., Roodbergen, J.A.F.: Aṣṭādhyāyī of Pāṇini. Sahitya Akadami, New Delhi (1998)
5. Macdonell, A.A.: Vedic Grammar. Karl J. Trunbar, Strassburg (1910)

[12] They include all the modern parts of speech.

A Study towards Design of an English to Sanskrit Machine Translation System

Pawan Goyal and R. Mahesh K. Sinha

Indian institute of Technology, Kanpur, India
pawang.iitk@gmail.com, rmk@iitk.ac.in

Abstract. We are experimenting to examine how AnglaBharati system designed to translate English to Indian languages could be adapted for translation to Sanskrit. The main contribution of our work is demonstration of machine translation of English to Sanskrit for simple sentences based on PLIL generated by AnglaBharati and *Aṣṭādhyāyī* rules. Presently our translation system caters to affirmative, negative, interrogative, imperative, active and passive voice sentences. In our study, we have selected a set of nouns and verbs that represent different semantic categories besides a few adverbs and adjectives. We anticipate using a number of Sanskrit resources on *Aṣṭādhyāyī* and morphological synthesis [[2]].

1 Introduction

Natural language is attributed to human intelligence, and equipping machines with natural language processing capabilities is one of the difficult tasks. Machine translation (MT) is one such task that requires inputs from multiple disciplines such as computational linguistics, psycholinguistics, cybernetics, cognitive science, computer science, artificial intelligence, etc. It is not enough to build knowledge bases embodying what is needed for analysis and synthesis of natural languages; we also need to understand which knowledge needs to be invoked at what stage. Building a fully automatic machine translation system with human competence is still a distant reality. However, MT systems with limited capabilities and machine aids for translation have been developed and are in widespread use. The present work on English to Sanskrit MT is attempt in this direction. MT has been a subject of investigation right from the time computers came into existence. As the problem is complex, different approaches have emerged that could be broadly classified as follows:

Direct substitution: Machine translation can use a method based on dictionary entries, which means that the words and phrases will be translated as they are by using a dictionary. It needs listing all the words and their clusters. Obviously, this method cannot cope up with the ambiguities and complexity of natural language.

Rule-based/knowledge-based/transfer-based: Under this category, a wide variety of systems have been developed. The basic method is to obtain a parse of the source language sentence with the help of a grammar and/or knowledge base. Then this parsed structure is transformed to a target language structure that is used to synthesize the target language sentence. Rules always suffer from being inadequate, providing limited coverage and also sometimes inappropriately producing some garbage.

G. Huet, A. Kulkarni, and P. Scharf (Eds.): Sanskrit CL 2007/2008, LNAI 5402, pp. 287–305, 2009.

Interlingual Machine Translation: In this approach the source language is first translated into an intermediate language which may be an artificial language embodying all the information of the source language in disambiguated form. The translation to the target language is obtained by transforming this intermediate structure into target language text. This approach is very attractive for developing a multilingual translation system with minimal additional effort. However, success of the approach very much depends upon the 'goodness' of the intermediate structure obtained. Several compromises are made to make it a workable model for a group of languages. PLIL (Pseudo Lingua for Indian languages) is one such intermediate structure that we have used for translating from English to Indian languages.

Corpus-based Machine Translation (CBMT): Availability of language corpora in electronic form opened up a variety of applications in NLP. Unilingual corpus is used for language modeling and probabilistic parsing. Example-based MT (EBMT) and Memory-based MT (MBMT) use parallel aligned bilingual corpora. Translation is obtained by matching the stored examples with the input. Statistical machine translation (SMT) also uses parallel corpora using a statistical modeling of translation at different levels. The success of these methods is obviously dependent upon availability of representative parallel corpora with wide and adequate coverage in the domain of application.

As we do not have parallel corpora for English and Sanskrit languages, using any corpus-based method was ruled out. The AnglaBharati MT system used a pseudo-interlingual approach for translation from English to Indian languages, and it was possible to study English to Sanskrit machine translation with minimal effort. This motivated us to use AnglaBharati system. Here the input English sentence is transformed to a pseudo-interlingua structure called PLIL (Pseudo Lingua for Indian Languages) using a pattern-directed rule-base, which is like a context-free grammar (CFG). PLIL is a structure that has the word order of a group of Indian languages and carries all the syntactic and semantic information needed for synthesizing the target Indian language text belonging to that group. A detailed description of the system can be found in the technical report ⟦1⟧. We directly use the PLIL obtained while translating a sentence from English to Hindi. Since the system recursively tries to unify all the possible patterns, at times it may fire multiple rules and yield multiple PLILs. A post editing is therefore required to choose the most appropriate parsing on the basis of context of the sentence. Once we get the PLIL, we need to build a text generator for the Sanskrit language to get the Sanskrit translation. Pāṇinian grammar has been used for building the text generator. The online resources provided by Huet ⟦2⟧ are used for generating the word forms.

1.1 Organization of the Paper

The paper has been organized as follows: Section 2 describes the basic methodology that we will be using in generating Sanskrit sentences from the PLIL obtained. Section 3 describes how we are generating morphosyntactic representation from the PLIL. Section 4 describes the rules for the disambiguation of *kārakas* and *vibhaktis*. Section 5 describes the generation of phonological output from the morphosyntactic representation. Section 6 describes the results obtained by us. Section 7 gives an insight into how

we can handle complex sentences through our mechanism. Section 8 discusses future work that is to be done.

2 From Pseudo Lingua Representation of English to Generation of Sanskrit Sentences

As discussed by Kiparsky [[4]], Pāṇinian grammar analyzes the sentences at a hierarchy of four levels of description, from semantic information to phonological output form. To understand this hierarchy, consider the English sentence with the output form as below:

‘A leaf falls from the tree’.

The derivation of the Sanskrit according the hierarchy is as follows:

Semantic information. This level encodes the ontology of the events and the speaker’s wish to express certain features of it (*vivakṣā*). This initial stage includes the following steps:

1. ***svatantraḥ kartā*. 1.4.54** (स्वतन्त्रः कर्ता)
   The participant which is independent bears the role of agent (*kartā*).
2. ***dhruvam apāye ’pādānam*. 1.4.24** (ध्रुवम् अपायेऽपादानम्)
   When there is separation, the participant which is fixed bears the source role (*apādāna*).
3. ***vartamāne laṭ*. 3.2.123** (वर्तमाने लट्)
   Roots denoting event that is happening in front of us are assigned present tense (*laṭ*).

Morphosyntactic representation

At this level, the arguments are assigned the thematic roles (*kārakas*) and the root is assigned abstract tense, thus we have:

1. *vṛkṣa* (वृक्ष) ‘tree’: source (*apādāna*)
2. *patra* (पत्र) ‘leaf’: subject (*kartā*)
3. *pat* (पत्) ‘fall’: present tense (*laṭ*)

Abstract morphological representation

vṛkṣa-ṅasi patra-su pat-tip

वृक्ष-ङसि पत्र-सु पत्-तिप्

‘tree-AblSg leaf-AccSg fall-PrAct3Sg’

Where, the first line shows the morphological representation in Sanskrit, and the second line shows that in English. ‘AblSg’ is ‘ablative singular’, for which the pratyaya used in Sanskrit is *ṅasi*, ‘AccSg’ is ‘nominative singular’, for which the pratyaya used in Sanskrit is *su*, ‘PrAct3Sg’ is ‘Present tense, active voice, third person, singular’ for which the pratyaya used in Sanskrit is *tip*.

Phonological output form

vṛkṣāt patram patati.[1]

[1] We do not implement interword sandhi.

वृक्षात् पत्रम् पतति.
'A leaf falls from the tree'.

In view of this hierarchy, the basic steps in translating an English sentence to Sanskrit sentence are:

1. Extract the semantic information of the sentence from the parse tree produced by the Angla Bharati system.
2. Assign thematic roles to the arguments and thematic roles to the root. (Morphosyntactic representation)
3. Assign pratyayas corresponding to morphosyntactic representation and generate the phonological output.

However, as discussed by Scharf ⟦3⟧, there is an overlap between the levels and we cannot draw an exact boundary. In our current implementation, however, we go by the hierarchy, given by Kiparsky. The system can be represented by figure 1.

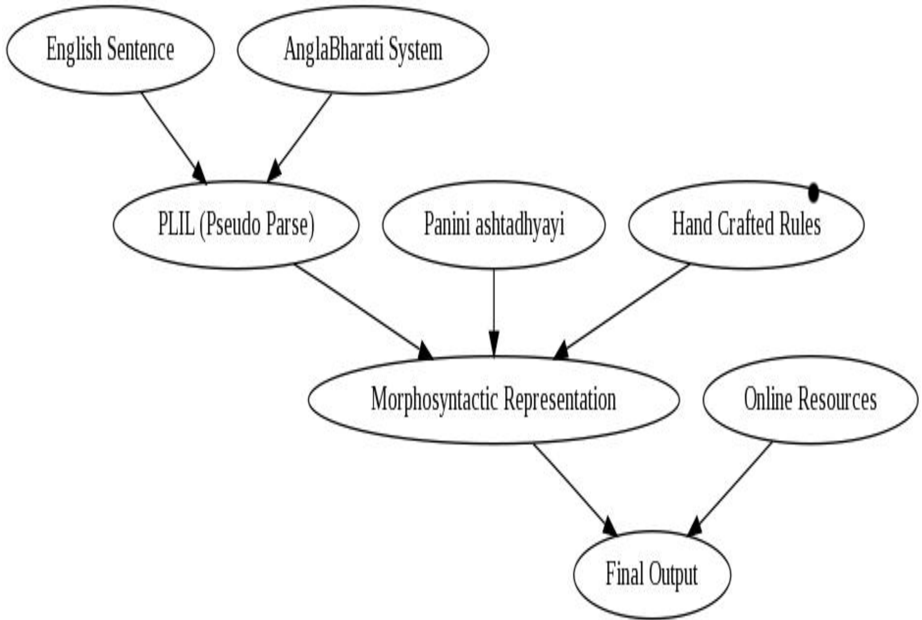


Fig. 1. System description

Let us explain the procedure through a simple example.

– The English sentence:
  He goes to market.
– The PLIL representation produced by AnglaBharati system:

```
<aff {sub_np ( he noun masculine singular third [human] [vaha:m
 8] [] [] ) } {pp (market noun neuter singular third [place]
[bAjZAra:m 6] [] [] ) ( to prep [ to ] ) } {main_vp_active ( go
 verb_2 normal normal masculine singular third [jA] 3 []  []
 ) } > . sviram
```

The PLIL representation is in a tree form. The root node is 'aff', which corresponds to affirmative sentences. Depending upon the sentence, various values are given to this root node, such as: **qs** - for interrogative sentences, **imp** - for imperative sentences, **let** - for sentences starting with let and **aff** - for affirmative sentences.

The root node has several children corresponding to phrases in the word. In the above tree, there are three children - 'sub_np', 'pp' and 'main_vp_active'. Some of the possible values that the children of each node can take are the following:

- 'sub_np' corresponding to a subject noun phrase
- 'pp' corresponding to a prepositional phrase
- 'obj_np' corresponding to a object noun phrase
- 'main_vp' corresponding to main verb phrase. To represent the voice, 'active' and 'passive' are also added to this child.

Let us traverse the children one by one as shown in the PLIL:

- 'sub_np': The different attributes that this child has stored are 'he noun masculine singular third [human] [vaha:m 8] [] []'; this represents how the information for the nouns has been coded. The attributes can be characterized as follows:
  1. 'he' - corresponds to the root word in English corresponding to the subject.
  2. 'noun' - is the category of the word.
  3. 'masculine' - stores the gender of the noun.
  4. 'singular' - stores the number of the noun.
  5. 'third' - stores the person information of the noun.
  6. '[human]' - stores the semantic category of the noun.
  7. '[vaha:m 8]' - it stores the corresponding representation in Hindi. 'vaha' is the Hindi word for 'he', 'm' stores the gender of 'vaha', that is 'masculine'. '8' is the paradigm number which is passed for the text generator for the generation of final form.
- 'pp': Prepositional phrase will have a preposition in it, which is shown in brackets in PLIL as '(to prep [to])'. Other attributes are same as that of 'sub_np'.
- 'main_vp_active': It stores attributes corresponding to a verb. For the example, the attributes are
  `go verb_2 normal normal masculine singular third [jA] 3 []  []`
  which can be characterized as:
  1. 'go' - corresponds to the root word in English corresponding to the verb.
  2. 'verb_2' - It shows the category 'verb' as well as the paradigm in English. Verbs have been characterized in 5 paradigms according to the endings. For example, for the root verb 'eat', the paradigm numbers and the form of the verb are: verb_1 - eat, verb_2 - eats, verb_3 - ate, verb_4 - eaten and verb_5 - eating.
  3. 'normal' - will store if the verb has been used in 'command' or in a sentence starting with 'let'.

4. 'normal' - this is the second 'normal'; this corresponds to some extra word(s) used in verb phrase such as 'has', 'have' etc.
5. 'masculine singular third' - will store the 'gender person number' information (crucial for verb conjugations in Hindi) corresponding to the noun, depending on which the form of the verb conjugates.
6. '[jA] 3' - it stores the corresponding representation in Hindi. 'jA' is the Hindi word for 'go', '3' is the paradigm number which is passed for the text generator for the generation of final form.

From the above representation, we can clearly see that the lexical database used has the corresponding entries of Hindi words. We need to substitute corresponding Sanskrit words in the lexical database in place of Hindi words. This database keeps most of the syntactic, semantic and other information for the need of disambiguation. For example, corresponding to 'he', the database retains the semantic information that it represents a 'human'. Corresponding to 'market', the semantic information of 'place' has been retained. This information will be used for Sanskrit text generator too. Since we are working with a small vocabulary, we have separately enlisted the Sanskrit word corresponding to different English words.

– Transferring the structure to corresponding representation in Sanskrit:
The corresponding structure that we get is:
subject तत् nom sg mas
pp आपण acc sg neu prep to
verb गम् prim pr ac sg trd
The words written in devanagari are the root words corresponding to Sanskrit. Each of the line corresponds to a different NP, PP or VP or a separate word. Main reason for this structure is the fact that in Sanskrit, unless a word is indeclinable, it declines. Thus, in an NP, the adjective will decline in the same way as the corresponding noun and so on. Attempt is to keep most of the information for declining a particular word within a line, although there are complicacies which will be discussed. We will be explaining each of the term in next sections.
– From morphosyntactic representation to phonological output:
In AnglaBharati system, all the forms corresponding to a verb or noun have been stored in form of tables. Sanskrit is a language with a rich morphology. If we consider a simple verb like *khād* corresponding to English verb 'to eat', it can have many prayogas[2] and verb forms[3], each of which appear in 10 tenses. Again for each tense, it can have 9 different output forms corresponding to number and person, which gives around 1000 forms corresponding to the verb *khād*. These forms vary depending upon the class in which the verb appears and other phonological information. So, it is not a good idea to have a table corresponding to endings and use it to generate the morphology from root word. The approach given in the

[2] *prayoga* designates the type of usage of the verb. It is of three kinds,
- *kartṛ-prayoga*: when the termination on the verb denotes the agent,
- *karma-prayoga*: when the termination on the verb denotes the *karman*, and
- *bhāva-prayoga*: when the verb denotes just the action of the verb, in which case a default (third person singular) verbal termination is used.

[3] These are special suffixes, which change the meaning of the verb. *ṇijanta* (causative) and *sannata* (desiderative) are the examples.

Aṣṭādhyāyī is fairly complex in which you need to go through a large set of rules each time to generate the phonological output. For the work here, we are using the online resources (data banks) provided by Huet ⟦2⟧. These are resources for nouns, verbs and particles, which given the morphosyntactic representation, generate the final form. We briefly discuss on how we can generate these forms using an algorithm. From the representation above, we get the following output:
saḥ āpaṇam gacchati. (सः आपणम् गच्छति)
– Finally a post processing module is used to change the word order if needed.

3 Generating the Morphosyntactic Representation from PLIL

This step needs incorporating the linguistic knowledge of Sanskrit. Firstly, since the PLIL is in a tree form, we need to parse the tree. A simple java program helps to get the following information:

- A particular noun, whether is a subject, an object or is used in a prepositional phrase?
- The adjectives are put in the same phrase (so that they share the same information as that of noun).
- The preposition used with the noun, which helps in deciding the vibhakti.
- Any other words, which may be helpful in deciding the vibhakti.

Below is the output of the java program for the example under consideration:

```
sub_np (he noun masculine singular third [human])
pp (market noun neuter singular third [place])
to prep [ to ]
main_vp_active (go verb_2 normal normal masculine singular third)
```

Thus the java program extracts the information in the sentence, 'he goes to market': 'sub_np' includes only the word 'he', 'to market' is a 'pp' and 'go' is the main verb. We are left with the following tasks:

1. Translate the English root words corresponding to their Sanskrit equivalents.
2. Generate the mapping for the kāraka in the case of nouns and lakāra in the case of verbs.
3. Get the gender, number information for nouns and person, number, prayoga, and vācya information for verbs.

Let us elaborate on each of the aspects here:

3.1 Translating Root Words

For translating root words, we need to expand the lexicon. Since we are only testing on a small vocabulary, we are just storing the corresponding Sanskrit words. Thus we have:
he तत्, market आपण and go गम्
All the adverbs which do not decline are stored with corresponding Sanskrit translation. The nouns are stored with the corresponding genders and number[4] information.

[4] Number information is stored only where relevant. There are nouns such as *dāra, lājā, gṛha* which take only plural declensions.

3.2 kāraka and lakāra Mappings

We first discuss in brief about the kārakas used in Sanskrit. kārakas are related not to the form of the word but with the meaning, they are needed to describe a particular action. There are 6 kārakas, *kartā, karma, karaṇa, sampradāna, apādāna* and *adhikaraṇa*. To convey the meaning of a kāraka, affixes, termed *vibhakti* are applied to a nominal stem. For example, the prathamā vibhakti is used to denote the kartā kāraka, the dvitīyā vibhakti is used to denote the karma kāraka. Particular vibhaktis are used when some avyayas occur in the composition, for example the pañcamī vibhakti is applied in composition of the word *ṛte*. Based on the semantic information, first we need to determine the kāraka. After the kāraka is determined, the appropriate vibhakti will be applied to the stem. In a prepositional phrase, if we encounter some indeclinables (*avyaya*s) for which a particular vibhakti has been assigned, we will use that particular vibhakti. There are seven vibhaktis in forming a noun. Let us discuss in brief, how these vibhaktis are applied:.

prathamā: In Sanskrit, no noun is spoken without a vibhakti. This vibhakti tells the form of the word. When the vācya of the verb is active (as in the example obtained by '`main_vp_active`', this vibhakti is used to denote the subject. Thus, in the above example, we have 'तत्' as root word with 'prathamā' as the vibhakti. If the vācya is passive, then also the PLIL automatically maps the object to '`sub_np`' and we need not change the rule. For example, in the sentence 'The food was eaten by Ram', 'the food' is mapped to '`sub_np`' and is assigned prathamā vibhakti.

dvitīyā: This is employed mainly to denote the karma kāraka. If we get an '`obj_np`' in PLIL, it will mostly take this vibhakti. The preposition 'to' used in English corresponds to this vibhakti most of the times.

tṛtīyā: This is mainly used in karaṇa as well as kartā. When the vācya is passive, then tṛtīyā denotes the subject. For example, in the sentence, *tvayā khādyate* (by you is eaten), the subject is denoted by applying tṛtīyā to the subject *yuṣmad* (you). (There is another vācya called *bhāva* which does not appear in English, for example the sentence *tena gamyate*. Since *gam* (to go) is transitive, it does not take passive construct in English.) Generally tṛtīyā maps to the English preposition 'by'.

caturthī: This vibhakti is used mainly in sampradāna, that is, if something is given. It maps to the preposition 'for' in general.

pañcamī: This vibhakti is used mainly in apādāna, where there is a sense of separation.. For example, 'the leaf fell from the tree', in this sentence that which is still is apādāna and is denoted by pañcamī. It maps to the preposition 'from' in general.

ṣaṣṭhī: This represents the relation between two nouns, such as in 'Ram's brother', Ram will have the ṣaṣṭhī vibhakti. It maps to the preposition 'of' in general.

saptamī: It is used to denote the adhikaraṇa kāraka as well as in other senses.
There are many exceptions to the default mapping that we have discussed for the vibhaktis[5] which will be discussed in detail in later section.

[5] The mapping described is not strictly Pāṇinian but in the reference of how it is being implemented. Exceptions are being handled separately.

There are 10 different lakāras in Sanskrit. For our analysis, we are considering only the simple sentences and therefore, we come across only five of them. The following lakāras are considered in our analysis:

1. *laṭ* lakāra: denotes the present tense. We have the following mappings for *laṭ*, where each mapping is represented by 'PL' meaning, it is an extract from the PLIL, then there is an example for the kind of sentence(s) in which this mapping occurs:

```
PL: verb_1|2+normal+normal pr\footnote{Extract from PLIL,
    already discussed in section 2.} (Ex: he (goes) to market.)
PL: verb_5+normal+is|are pr (Ex: He (is going) to market.)
PL: verb_5+normal+has_been pr (Ex: He (has been going) to market.)
PL: verb_4+normal+is|are_being pr
(Ex: Food (is being eaten) by Ram.)
PL: verb_4+normal+is|are pr (Ex: Food (is eaten) by Ram.)
PL: verb_1+normal+wdoes|wdo pr (Ex: Does he go to market?)
```

   In the above file '|' represents the 'or', thus either verb 1 or verb 2 will lead to the *laṭ* lakāra represented as 'pr'. '+' is simply used to fill up the gap. In the example take, we have 'verb 2 normal normal', which has been taken here as 'verb 2+normal+normal'. We have given an example in the next line of each rule. The bracket encloses the relevant part.
2. *laṅ* lakāra: denotes the imperfect tense. We have the following mappings for *laṅ*:

```
PL: verb_4+normal+has|have im  (Ex: He (has gone} to market.)
PL: verb_3+normal+normal im  (Ex: He (went) to market.)
PL: verb_5+normal+was|were im  (Ex: He (was going) to market.)
PL: verb_4+normal+had im  (Ex: He (had gone) to market.)
PL: verb_4+normal+was|were_being im
Ex: The food (was being eaten) by Ram.
```

   The notations used are same as in *laṭ*. 'im' represents the 'imperfect tense'(*laṅ* lakāra).
3. *lṛṭ* lakāra: represents simple future tense. We have the following mapping:

```
PL: verb_1+normal+will|shall fut (Ex: He (will go) to market.)
PL: verb_5+normal+will_be fut (Ex: He (will be going) to market.)
```

   'fut' represents 'future tense' (*lṛṭ* lakāra).
4. *vidhiliṅ* lakāra: represents the optative mood. We have the following mappings:

```
PL: verb_1+normal+should op  (Ex: He (should go) to market.)
PL: verb_1+normal+has_to op  (Ex: He (has to go) to market.)
PL: verb_4+normal+should_have op
Ex: He (should have gone) to market.
PL: verb_1+normal+must op  (Ex: He (must go) to market.)
PL: verb_1+normal+may op  (Ex: He (may go) to market.)
```

   'op' represents 'optative mood' (*vidhiliṅ* lakāra).

5. *loṭ* lakāra: represents the imperative mood. We have the following mappings:

```
PL: verb_1+let+normal ip  (Ex: (Let) him (go) to market.)
PL: verb_1+command+normal ip  (Ex: (Go) to market.)
```

   'op' represents 'imperative mood' (*loṭ* lakāra).

There are other lakāras in Sanskrit but are not used in simple sentences. Above discussed 5 lakāras are mainly in use. We will discuss others when we will move to complex sentences.

3.3 Getting the Gender, Person Information in Nouns and Person, Number, Prayoga, vācya Information in Case of Verbs

Let us first see how this information is being extracted for nouns: Sanskrit has three genders:

1. Masculine (*puṁliṅga*)
2. Feminine (*strīliṅga*)
3. Neuter (*napuṅsakaliṅga*)

The problem in using the gender information from PLIL is that a word having 'masculine' gender in English may not have puṁliṅga in Sanskrit. For example, 'pot' in English has 'neuter' gender, but its Sanskrit equivalent *ghaṭa* is puṁliṅga. Similarly, 'story' in English has 'neuter' gender, but its Sanskrit equivalent *kathā* is strīliṅga. In Sanskrit, all the three genders decline in a different fashion and therefore it is crucial for us to have the gender information. On the other hand, the adjectives will take the gender of the noun. Therefore, while the nouns are stored with the information of gender, the adjectives are stored without this information. The pronouns have the same root word (*tad* for 'he', 'she' and 'it'). Therefore, the gender information from PLIL is passed to decline the form.

As far as the information about number is concerned, Sanskrit has three numbers (*vacana*):

1. singular (*ekavacana*)
2. dual (*dvivacana*
3. plural (*bahuvacana*)

Since English has only the singular and plural, dual information has to be extracted from the PLIL. For example, for the English sentence, 'he brought two mangoes from the market', we will get the number of 'mangoes' as plural. We are keeping track of the adjective 'two' which when precedes a noun, makes its number as dvivacana. In PLIL, 'you' always comes with the number 'plural', we therefore need to change it to ekavacana. When it comes to verbs, the information is extracted as follows:
In Sanskrit, there are three persons (*puruṣa*):

1. Third person (*prathama puruṣa*)
2. second person (*madhyama puruṣa*)
3. first person (*uttama puruṣa*)

The PLIL attaches this information to the verb, but it does not give the second person. When there is a verb with active voice with 'you' as its kartā (in imperative active, for example, "Go to your teacher tomorrow") , the PLIL gives the person as 'third'. We therefore check specially for the case when the verb in imperative active is corresponding to 'you' and change the person to 'second'. Other instances are fine.

The number information is also given in the PLIL, but as we saw in case of nouns, there are three number. The information for dual number needs to be extracted. In nouns, it was not a problem since a noun can have the number as dual only when there is an adjective 'two' preceding it, but verb can have this case whenever there are two nouns, for example in the sentence 'Ram and Shyam ate together', the verb was analyzed in PLIL as:

```
{main_vp_active(eat verb_3 normal normal neuter plural third
[KA] 1 [] [])}
```

Since the PLIL gives it a plural, we need to keep track of the number of nouns joined by 'and'. If the number is two, we assign dvivacana, if more than two, we assign bahuvacana. In case of 'you' being the kartā of the verb with the verb being in active voice, the PLIL gives its number as plural. We change it to ekavacana.

Regarding the vācya information, it is given in the PLIL as '`main_vp_active`' and '`main_vp_passive`' which is simply used as it is. Prayoga information will be discussed in complex sentences. We are considering in simple sentence only the primitive prayoga.

4 Disambiguation for kārakas and vibhaktis

As discussed in the section above, the mapping from PLIL to the morphosyntactic representation in Sanskrit seems to be fairly simple, but it is not so. When it comes to assigning kārakas, we need to keep track of many rules given in Pāṇini's *Aṣṭādhyāyī*. These rules assign kārakas depending upon the verbs used in the sentence as well as semantics of noun. There are rules which assign vibhaktis depending on context words. The rules require to keep track of the verb as well as context words. In the following description, we give examples of sentences in which mapping from English to Sanskrit is not straightforward; however, by using certain rules from the *Aṣṭādhyāyī*, we can get the correct translation. We will look at the rules for some of the kārakas and vibhaktis.[6] First, we look at the rules for disambiguation of kāraka.

4.1 Karma kāraka

We look at the following rules:

adhiśīṅsthāsāṃ karma. **1.4.46** (अधिशीङ्स्थासां कर्म)
Translation: That which is the site of the verbs *śī* 'to lie down', *sthā* 'to stand', *ās* 'to sit', when preceded by the preposition *adhi* is called *karma kāraka*.

[6] Some of the rules which need a detailed analysis, such as determining karmapravacanīya, are still not implemented.

Implication: When the above verbs are translated in Sanskrit with *adhi* preceding them, we look for the noun having 'place' in its semantics and assign it the 'karma' of the verb.[7]
Ex: Hari lies down in Vaikuntha. Here 'lies down' will be translated *śī* preceded by *adhi*. The word 'Vaikuntha' has the semantics 'place' and the rule applies. Note that the preposition 'in' maps in general to 'adhikaraṇa' but the rule maps it to 'karma' and we have the translation *hariḥ vaikuṇṭham adhiśete*.

4.2 Sampradāna kāraka

We look at the following rules:
karmaṇā yamabhipraiti sa sampradānam. **1.4.32** (कर्मणा यमभिप्रैति स सम्प्रदानम्)
Translation: The person whom one wishes to connect with the object of the verb *dā* (to give) is called *sampradāna*.
Implication: All the objects of the verb 'give' are given the *sampradāna* and for the sentences, 'he gives alms to the boy', 'boy' gets *sampradāna* by the rule and we have the Sanskrit sentence, *saḥ bālakāya bhikṣām dadāti*.
rucyarthānāṃ prīyamāṇaḥ. **1.4.33** (रुच्यर्थानाम् प्रीयमाणः)
Translation: In case of verbs having the signification of the root *ruc* (to like), the person or thing that is pleased or satisfied, is called *sampradāna*.
Implication: The rule is applicable not only to the root *ruc* but to roots having this signification. This allows us to have a list of these verbs (can be extracted from word net). These verbs give the one pleased the *sampradāna*, thus 'I like sweat-meat' is translated as *mahyam modakaḥ rocate*.
ślāghahnuṅsyāśapāṃ jñīpsyamānaḥ. **1.4.34** (श्लाघह्नुङ्स्थाशपां ज्ञीप्स्यमानः)
Translation: In case of verbs *ślāgh* (to praise), *hnu* (to take away), *sthā* (to stand), and *śap* (to curse), the person whom it is intended, is called *sampradāna*.
Implication: It again takes care of variety of sentences, for example, 'he waits for me', here 'wait' gets translated as *sthā* and 'I' gets the *sampradāna*. Thus we have the Sanskrit sentence *saḥ mahyam tiṣṭhate*. Similarly any sentence, for which any of these Sanskrit root word is employed, is taken care by this rule.
spṛher īpsitaḥ. **1.4.36** (स्पृहेरीप्सितः)
Translation: In case of verb *spṛh* (to desire), the thing desired is called *sampradāna*.
Implication: Consider the sentence, 'he desires flowers', the root 'desire' helps us to assign *sampradāna* to 'flowers' giving us the translation *saḥ puṣpebhyaḥ spṛhayati*.
krudhadruherṣyāsūyārthānāṃ yam prati kopaḥ. **1.4.37** (क्रुधद्रुहेर्ष्यासूयार्थानां यम् प्रति कोपः)
Translation: In case of verb having the sense of *krudh* (to be angry), *druh* (to injure), *īrṣyā* (to envy), *asūyā* (to detract), the person against whom these feelings are directed is called *sampradāna*.
Implication: This again refers to a wide variety of sentence. It is dealt with in the same way as 1.4.34.

[7] In Pāṇinian grammar, which begins with meaning conditions rather than with a sentence in another language, it is properly the semantic object, rather than a speech form in English or Hindi, that is assigned a kāraka term. In the current example, it is the place rather than a word denoting the place that would be termed *karman*. Likewise throughout. (Editor's note).

4.3 Apādāna kāraka

We look at the following rules:
***bhītrārthānāṃ bhayahetuḥ.* 1.4.25** (भीत्रार्थानां भयहेतुः)
Translation: In case of words implying ‘fear’ and ‘protection from danger’, that from which the danger or fear precedes is called *apādāna*.
Implication: Thus for the sentence ‘he is afraid of the thief’, ‘the thief’ is the one from which fear precedes and gets the *apādāna*. Similarly for ‘he protects from the thief’, ‘the thief’ gets the *apādāna*. Thus the translation being, *saḥ corāt trāyate* | *rakṣati*.
***bhuvaḥ prabhavaḥ.* 1.4.31** (भुवः प्रभवः)
Translation: The source of the agent of the verb *bhū* (to become) is called *apādāna*.
Implication: It again refers to catch certain structures in the English sentence such as ‘... from a (place)’ etc. Even if the verb *prabhava* is used, we can apply this rule. Thus for the sentence, ‘The Ganges flows from the Himalayas’, we have the translation *gaṅgā himavataḥ prabhavati*.

Let us look at some of the examples for the disambiguation of *vibhakis*.

4.4 dvitīyā vibhakti

***antarāntareṇa yukte.* 2.3.4** (अन्तरान्तरेण युक्ते)
Translation: A word joined with the word *antarā* or *antareṇa* takes the second case affix.
Implication: *antareṇa* and *antarā* fall into the category of *avyaya* in Sanskrit. The word *antareṇa* is used for ‘without’ and *antarā* for ‘between’. Thus for the sentence, ‘There can be no happiness without Hari’, if we use for ‘without’ the Sanskrit word *antareṇa*, the rule applies and we have *hari* with ‘karma’. For using such rules, we first assign all the root words and the mapping of kāraka is done later.
***karmapravacanīyayukte dvitīyā.* 2.3.8** (कर्मप्रवचनीययुक्ते द्वितीया)
Translation: The second case-affix is employed after a word which is joined with a karmapravacanīya.
Implication: We need to determine whether a word having been assigned the term *karmapravacanīya* is joined with a particular word. *karmapravacanīya* is also defined in *Aṣṭādhyāyī*. We take a rule for example: *tṛtīyārthe.* 1.4.85
Translation: The word *anu* is termed *karmapravacanīya* when it has the force of the third case. Now consider the English phrase, ‘the army lying along side the river’, here ‘along side’ has a force of third case and it maps to Sanskrit wrd *anu*. From the rule, *anu* gets the term *karmapravacanīya* and the river will get the dvitīyā by the rule 2.3.8, which otherwise could have got *tṛtīyā* by the force of third case. Thus we have the Sanskrit traslation as *nadīm anu avasitā senā*.
***kālādhvanorattantasaṃyoge.* 2.3.5** (कालाध्वनोरत्तन्तसंयोगे)
Translation: After a word denoting time or length, *dvitīyā* is employed when denoting full duration.
Implication: Consider the English sentence, ‘he reads for a month’, ‘a month’ denotes a full duration and is assigned *dvitīyā* by the above rule giving us *māsam adhīte*.

4.5 tṛtīyā vibhakti

***sahayukte'pradhāne.* 2.3.19** (सहयुक्तेऽप्रधाने)
Translation: When the word *saha* is joined to a word, the latter takes the third case, when the sense is that the word in third case is not the principal.
Implication: Consider the English sentence, 'He went with his brother.', the preposition 'with' is translated as *saha* in Sanskrit and the rule applies putting 'brother' in *tṛtīyā vibhakti* and we have the translation, *saḥ tasya bhrātrā saha agacchat*.
***yenāṅgavikāraḥ.* 2.3.20** (येनाङ्गविकारः)
Translation: By whatsoever limb, being defective, is pointed out the defect of the person, after that the third case affix is employed.
Implication: Consider an English sentence, 'He is blind of one eye', general mapping gives 'eye' the *ṣaṣṭhī*, but the rule picks the term 'blind' as the defect and gives 'eye' the third case affix, giving the translation, *saḥ akṣṇā kāṇaḥ*.

4.6 Caturthī vibhakti

***namaḥ svastisvāhāsvadhālaṃvaṣaṅyogācca.* 2.3.16** (नमः स्वस्तिस्वाहास्वधालंवषड्योगाच्च)
Translation: The fourth case affix is employed in conjunction with words *namaḥ* (salutation), *svasti* (peace), *svāhā, svadhā* (used in oblations), *alam* (sufficient for) and *vaṣaṭ* (a term of oblation).
Implication: This rule is differnt from previously told for this kāraka since deals with the context words. Thus for 'salutation to Gods', the translation would be *svasti prajābhyaḥ*.

4.7 ṣaṣṭhī vibhakti

***adhīgarthadayeśāṃkarmaṇi.* 2.3.52** (अधीगर्थदयेशांकर्मणि)
Translation: Of the words having the sense of *adhi* (remembering), *daya* (to pity) and of *īśa* (to rule), the object takes the sixth case affix.
Implication: For the sentences such as 'he remembers the mother', the object 'mother' will take the *ṣaṣṭhī vibhakti* and we will have the form *mātuḥ smarati*.

4.8 Saptamī vibhakti

***āyuktakuśalābhyām cāsevāyām.* 2.3.40** (आयुक्तकुशलाभ्याम् चासेवायाम्)
Translation: In conjunction with the words *āyukta* (engaged), *kuśala* (skillful), when meaning entire absorption in an engagement, the *saptamī vibhakti* is used after a word.
Implication: The *saptamī vibhakti* will be applied to the nominal stem that comes in conjunction with these words.

Above discussed are the rules which can be used for kārakas and vibhaktis disambiguation for majority of the cases. However, there are other rules for finer distinction. One can refer to these rules in Pāṇini's *Aṣṭādhyāyī* [[5]].

5 From Morphosyntactic Representation to Phonological Output

As discussed in the previous sections, we are able to represent the English sentence into its morphosyntactic representation in Sanskrit. Now the data resources provided by Huet are used for generating the final form of each of the word. The resources are in the form of XML trees. The tree is parsed using a java program. We have characterized the input of the tree in the following terms with the all possible values that the tree can take:

1. verb form: It takes the values depending upon the usage of the verb, whether it is used as causative, primitive or desiderative. The tree has entries corresponding to:
   (a) prim - primitive sense, the normal *lakāra* forms
   (b) ca - causative *ṇijanta*
   (c) des - desiderative *sananta*
2. lakāra: It takes the values corresponding to the tenses. Tree has following eight *lakāra*s:
   pr - present tense, *laṭ lakāra*, **im** - imperfect tense *laṅ lakāra*
   fut - future tense, *lṛṭ lakāra*, **op** - optative mood *vidhiliṅ lakāra*
   ip - imperative mood, *loṭ lakāra*, **pef** - periphrastic future tense *luṭ lakāra*
   prf - perfect tense, *liṭ lakāra*, **aor** - aorist tense, *luṅ lakāra* We need only the first five for our analysis.
3. prayoga & *pada*: It takes the values corresponding to the *pada* as well as *prayoga*. In Sanskrit, the verbs are categorized as to whom the fruit of the action belongs. If the fruit belongs to the doer, the verb is *ātmanepadī*, if fruit belongs to another, it is *parasmaipadī* and if it can belong to both depending upon the sentence, the verb is *ubhayapadī*. The *ātmanepadī* and *parasmaipadī* verbs conjugate in a completely different way and therefore in the tree, each root has been stored according to both of the forms. We need to input which form we are looking for. Also, if we want a passive construct, we can give in this input (because all verbs conjugate only uniquely in passive construct (A single verb has only one set of declensions.).)
   Thus, this input accepts the following:
   (a) ac - *parasmaipada, kartṛ prayoga*
   (b) at - *ātmanepada, kartṛ prayoga*
   (c) ps - passive voice, *karma prayoga*
4. vacana: This can accept one of the inputs for number (Section 3.3)[8] item *puruṣa*: This can accept one of the inputs for person (Section 3.3)[9]
5. root: accepts the *dhātu*(should not have the *upasarga* attached to it.)

Let's look at an entry corresponding to verb tree.

```
<?xml><!DOCTYPE forms SYSTEM "flexed.dtd">
<forms><f form="agacCat"><v><cj><prim/></cj><t><cj><im/><ac/></cj>
</t><np><sg/><trd/></np></v><s stem="gam"/></f>
```

[8] 'sg' for singular, 'du' for dual and 'pl' for plural.
[9] 'trd' for third person, 'snd' for second person and 'fst' for first person.

We are looking for the information of 'form' corresponding to a particular 'stem (root)'. In the tree, we need to give the input of 'prim', 'im', 'ac', 'sg', 'trd' and '`stem = "gam"`'. The extracted information (section 3) is passed to the XML tree parser to get the 'form' corresponding to a given root.

Similarly, the inputs to a noun tree are characterized as follows:

1. vibhakti: these are discussed previously. Tree has the following vibhakti:
   nom - nominative, *prathamā*, **acc** - accusative, *dvitīyā*
   ins - instrumental, *tṛtīyā*, **dat** - dative, *caturthī*
   abl - ablative, *pañcamī*, **gen** - genitive, *ṣaṣṭhī*
   loc - locative, *saptamī*, **voc** - vocative, *sambodhana*
2. vacana: Same entries as discussed in verb XML tree.
3. liṅga: there are three possible inputs for gender (Section 3.3) [10]
4. prātipadika: We will input a particular noun which we need to get the declined form for.

Now, let us look what a tree for 'noun' forms look like:

```
<?xml><!DOCTYPE forms SYSTEM "flexed.dtd">
<forms><f form="rAmAn"><na><acc/><pl/><mas/></na><s stem="rAma"/></f>
```

In the tree, we need to give the input of 'acc', 'pl', 'mas' and '`stem="rAma"`'. In this way, we are able to get the form for a particular noun and a particular verb using the Angla Bharati system and the data banks provided by Huet. There are certain issues, however that we must keep track of, which help us to handle the verbs with *upasarga* and missing noun in databank:

verbs with *upasarga* - *sopasarga dhātu*: In Sanskrit, by using *upasarga*, a *dhātu* can represent various meanings. Foe example, the verb *vad* - 'to tell', when preceded by *apa* gives the sense of 'to blaspheme' as *apavadati*, when preceded by *vi* gives the sense of 'to quarrel' as *vivadati* and so on. Thus if we have a root verb as upasarga+dhātu, we pass the dhātu to the XML tree and finally, the two parts are joined using a *saṃdhi* module.

missing noun - The number of nouns is unlimited, but we can still deal with this in certain situations. If the semantics of noun is 'place', 'post holder' etc. where we can accept the same forms as in Hindi, we pick that form from the PLIL and decline using the last vowel. That is, we look for a noun having the same last vowel and use its declined form with changing the initial letters.

6 Results

We have tested our system implementation translating affirmative sentences without mood, negative sentences, interrogative sentences, modal sentences and passive constructions. The Sanskrit translations obtained are found to be quite satisfactory. In some cases, it is possible to produce a more acceptable translation using other mechanisms. Some examples are given below:

[10] 'mas' for masculine, 'fem' for feminine and 'neu' for neuter.

1. He was going to market. ⇒ सः आपणम् 'अगच्छत्' (गच्छन् आसीत्)
2. He had gone to market. ⇒ सः आपणम् अगच्छत् (गतवान्)

The words shown in brackets give an alternate translation, which is more acceptable for the words in inverted commas, for example, सः आपणम् गच्छन् आसीत् is more acceptable Sanskrit translation for the sentence, 'he was going to market' than the translation सः आपणम् अगच्छत्, which is produced by the system. Since we have not analyzed the participles, we give a good enough translation. The other sentences with 'perfect continuous' aspect also need the use of participles. In the above sentences, we manually picked up the correct PLIL in case there were many options available. Let's look at some other variety of sentences that the program is able to handle.
3. He drank water from the pot. ⇒ सः घटात् जलम् अपिबत्
4. Ram went to the market with Shyam. ⇒ रामः श्यामेन सह आपणम् अगच्छत्
5. He told story among the friends. ⇒ सः मित्राणाम् मध्ये कथाम् अकथयत्

Imperative and optative mood has been covered by the system, for which we are giving a few examples below:
6. He may | (has to) | must go to market. ⇒ सः आपणम् गच्छेत्
7. He might have gone to market. ⇒ सः आपणम् अगच्छत् स्यात्
8. Go to your teacher tomorrow. ⇒ तव गुरुम् श्व गच्छ
9. He should have gone to market. ⇒ सः आपणम् गच्छेत्
10. Let him go to market. ⇒ सः आपणम् गच्छतु

Thus some of the sentences get translated by adding some extra word. For example, the sentence 'He could have gone to market' got translated as 'सः आपणम् अगच्छत् सम्भवेन'. The word 'स्यात्' has been added while matching the verb phrase. For the verb phrase, `verb_4 normal could_have`, we store the tense information as `im+`संभवेन, where 'संभवेन' refers to the extra word to be added in the Sanskrit translation. Other sentences got translated by using *loṭ* and *vidhiliṅ*.
Passive constructs were also dealt with nicely by the system, for example:
11. What was eaten by you? ⇒ त्वया किम् अखाद्यत ?

7 Handling Complex Sentences

This section deals with the kind of sentences that we have not yet incorporated in the system. However, we plan to do so using a Pāṇinian framework. We discuss the complexities involved that need to be taken care of.

– In Pāṇinian grammar, causatives are dealt with by applying the pratyaya *ṇic* to a root and then conjugating the derived form. However, in English, verbs such as 'had', 'got' are used for causative sentences. Therefore, the mapping is not straightforward. To take care of these sentences, we need to take clue from PLIL, as shown in the following example:

12. English sentence: I got my hair cut.

```
PLIL: <aff {sub_np ( i noun dont_care singular first [human] [mEM:m 8] []
 [] ) } {obj1_np ( my adjective any [poss_case] [apanA/merA] [] [] )
 (hairs noun neuter plural third [body_part] [bAla:m 6] [] [] ) }
 k1 {main_vp_active ( gotcutverb_3 normal normal dont_care singular
 first[katavA] 1 [] [] ) }> .sviram
```

Sanskrit Translation: अहम् मम केशानि अकर्तय
From the PLIL, the information regarding ‘causative’ use has been encoded as ‘got’ preceded by cut in the first sentence. It can be used to find whether a sentence is in causative or not. The hint will be passed to morphosyntax to get the verb form as ‘causative (‘ca’ in the XML tree)’.

– In Pāṇinian grammar, desideratives are dealt with by applying the pratyaya *san*. However, in English, these senteces have the structure of ‘.. want(s) to ‘verb’ ...’. The verb ‘want’ represents the desire to do the action that ‘verb’ says. The mapping is not direct here. Below is an example of a desiderative English sentence and what kind of processing is needed.
13. English sentence: I want to see the garden.

```
PLIL: <aff {sub_np ( i noun dont_care singular first [human]
[mEM:m 8] [] [] ) } {toinf ( the det[] [anda] [A] ) ( garden noun
neuter singularthird [place] [bAga:m 6] [] [] ) k4 ( see_1
verb_1 non_finite to masculine singular third [xeKa] 11 [] []
 ) to_in } {main_vp_active (want verb_1 normal normal
dont_care singular first [cAha] 11 [] [] ) } > . sviram
```

Sanskrit Translation: अहम् उद्यानम् दिदृक्षामि
In PLIL, the information is given by ‘want’ being analyzed as main verb and ‘see’ being analyzed as the ‘verb’. Thus ‘see’ will be passed to the morphosyntax with the ‘desiderative’ form. This form is stored in the XML tree in the node ‘verb forms’ as ‘des’, that is how it will be passed.

– Gerunds are indeclinable derivative forms from verb root to signify a ‘preceding action’. In Sanskrit, these forms are translated using the pratyaya *ktvā*. However, which of the actions is preceding needs to be determined. For example, consider the sentence, ‘having read, he sat on the chair’. In this sentence, action of ‘sitting’ starts after the action of ‘reading’. However, the mechanism to determine the sequence of actions from PLIL is not yet incorporated in the system.

– An infinitive expresses a subordinate action which is the goal, purpose or reason for the main action. In the current implementations, these are being taken care of by using *caturthī vinhakti*. For example, if we take the sentence, ‘ram goes home to drink water’, the phrase ‘to drink’ is the infinitive. This is translated in Sanskrit using the pratyaya *tumun*.

8 Conclusion

The current work is in the direction of building an English to Sanskrit Machine Translation system. Our aim has been to examine how the semantic information can be passed

through the Pāṇinian grammar resulting in appropriate Sanskrit form. We have shown some of the outputs for the simple sentences taken for translation. The system is at its base level. We are working on the algorithms for incorporating larger variety of sentences. We need to develop the lexicon appropriately by entering the Sanskrit words in place of Hindi words. Also, we need to incorporate rules from Pāṇinian grammar for disambiguation. The resources that we are using are prone to errors and we need to build our own text generator. Work has been going on in this direction and algorithms have been identified. Also, since we are successful in building a text generator for Sanskrit from PLIL, and PLIL to Hindi text generator already exists. We would also like to examine Machine Translation from Hindi to Sanskrit language. The study will also give us direction for translating Hindi to Sanskrit without going through PLIL.

Acknowledgements

We acknowledge our gratitude to Dr. Gerard Huet for providing the online databanks which were very helpful for building the text generator.

References

1. Sinha, R.M.K.: A Pseudo Lingua for Indian Languages for Translation from English. Technical Report, Language Technology Lab, Department of Computer Science and Engineering, Indian Institute of Technology, Kanpur (2004)
2. Huet, G.: Sanskrit Linguistic Resources, http://sanskrit.inria.fr/DATA/XML/
3. Scharf, P.: Modeling Pāṇinian Grammar. In: Huet, G., Kulkarni, A. (eds.) Proceedings of the First International Sanskrit Computational Linguistics Symposium, pp. 77–94 (2007)
4. Kiparsky, P.: On the Architecture of Pāṇini Grammar. Lectures delivered at the Hyderabad Conference on the Architecture of Grammar, January 2002, and at UCLA, March 2002 (2002)
5. Vasu, S.C.: The aṣṭādhyāyī of Pāṇini. Motilal Banarasi Das 1 (1962)

Phonological Overgeneration in Paninian System

Malhar Kulkarni*

Indian Institute of Technology, Mumbai, India
Vāsudeva Śāstrī Abhyankar Pāthaśālā, Pune, India
malhar@iitb.ac.in
http://www.hss.iitb.ac.in

Abstract. In this paper an attempt is made to study the problem of overgeneration that is caused by the application of the system of *Pāṇini* The system of *Pāṇini* is made up of certain rules stated by him and his commentators namely, *Kātyāyana* and *Patañjali*. These rules are supposed to produce the forms that are used in the language, i.e. Sanskrit. However, sometimes the technical application of these rules produces such forms which are not actually used in the language. In fact, sometimes it is beyond human capacities to use such forms. In the present paper two such cases dealing with the phonological overgeneration are studied and possible solutions are proposed to avoid the problem.

1 Introduction

It has been demonstrated by Kiparsky and Staal (1988) how Paninian system functions on four levels, namely, semantic, deep structure, surface and phonological. This system however sometimes over-generates in perhaps, some of these levels. Of course *Pāṇini* (P) has no doubt laid down certain constraints with the help of which the system produces supposedly un-over-generated forms. Prince and Smolensky (2004), have devoted a section on Panini's theorem of constraint ranking (5.3) Of course our judgement regarding the over-generativeness of a rule in the *Aṣṭādhyāyī*(A), it must be admitted here, is based entirely upon whatever evidence in the form of pre-paninian literature available to us.

2 Phonological Over-Generation

This paper is devoted to phonological over-generation that still happens with all the possible constraints applying. There are two aspects that are studied in this paper,

(1) Nasalization and (2) Phonetic doubling

* I wish to express gratitude to my students Ms.A.Ajotikar, Ms.T.Aajotikar and Ms.Sarnaik, at the Abhyankarashastri Pathashala, Pune for helping me type out the forms in tables presented in this paper. I also wish to express my gratitude to all the scholars who made suggestions and remarks which helped improve this article immensely. I wish to thank Prof. Kiparsky for providing me with the details of the reference of one of his forthcoming publications.

G. Huet, A. Kulkarni, and P. Scharf (Eds.): Sanskrit CL 2007/2008, LNAI 5402, pp. 306–319, 2009.

2.1 Nasalization

8.4.45 states that *yar*[1] occurring at the end of a *pada*, is optionally, (*preferably*, according to Kiparsky 1980:1) substituted by the nasal, if a nasal follows.
(1) *etad murāriḥ*
= *etan murāriḥ* / *etad murāriḥ* ... 8.4.45

Kātyāyana(K) has added a *Vārttika*(V) on this rule, to the effect that this nasalization takes place permanently if the following nasal is a part of a suffix
(2) *tad* + *maya*
= *tan-maya* ... 8.4.45 + K's V.
= *tanmaya*

Environment for nasalization: However, if we look at the way P has stated this rule, we have to take into account following table which shows clearly all possible environments in which this rule should apply and the possible results in the form of substitution of a nasal consonant. The top row and the left column, in the table, show the possible environment. The bottom row shows the resultant nasal consonant in place of the consonant written in the same column in top row. Thus for instance,

[..y] + [n..] / [m..] / ... 8.4.45 + K's V
= [..y#] + [n..] / [m..] /

Table 1. Consonants and their substitutes according to 8.4.45

	y	v	r	l	ñ	m	ṅ	ṇ	n	jh	bh	gh	ḍh	dh	j	b	g	ḍ	d	kh	ph	ch	ṭh	th	c	ṭ	t	k	p	ś	ṣ	s
ṅ																																
ñ																																
ṇ																																
n																																
m																																
	y	v	ṇ	l	ñ	m	ṅ	ṇ	n	ñ	m	ṅ	ṇ	n	ñ	m	ṅ	ṇ	n	ṅ	m	ñ	ṇ	n	ñ	ṇ	n	ṅ	m	ñ	ṇ	n
	#	#	*	#																										*	*	*

Table 1 shows that any consonant mentioned in the top row occurring at the end of a *pada* and followed by any of the nasal consonants mentioned in the left hand column, is substituted by the nasal consonant shown in the bottom row. # mark is used to show the nasal feature in the bottom row. * shows that these substitutions are not attested in Sanskrit. The order of sounds followed by P in his *pratyāhāra sūtra*s is maintained here.

There are certain sounds in this table which are directly not applicable for this operation as they never occur at the end of a *pada* in Sanskrit. Such sounds are- *y, l, ṅ, ñ , jh, bh, gh, ḍh,dh, kh, ph, ch*. Some grammatical entries do end in some

[1] These phonemes are- all the stops including the nasals, semi vowels(*y, r,l,v*) and sibilants except *h*.

of these sounds and hence it can be argued that by applying operations related to 0 suffix, one can generate *pada*s with these sounds at the end. However, this argument does not hold valid as in the case of these consonants, the other rules namely, 8.2.30, 8.2.39 etc. will substitute them with the other consonants.

Thus consider the following example:

(3)	*gumph*		... *Dhātupāṭha* 6.31
	gumph	+ *kvip*	... 3.2.178
	gu ph	+ *kvip*	... 6.4.24
	gu ph	+ 0	... 6.1.67
	= *guph*		
	guph	+ *su*	... 4.1.1,2
	guph	+ 0	... 6.1.68
	gub		... 8.2.39
	gub/gup		... 8.4.56
	= *gub / gup*		

In the same way, other consonants will be substituted.

Overgenerated nasalization: Now the rule, applied to all the remaining consonants should also apply to the following example:

(4)	*catur mukha*	...... 8.4.45
	catu ṇ mukha	
	= *catu ṇmukha*	

However, this resultant form is not acceptable in Sanskrit. This is clearly an overgeneration. 8.4.58 states the substitution of a nasal in place of an *anusvāra* when followed by almost same consonants (called as *yay* by P) mentioned in the top row of Table 1 above except the last three. The rule can be shown as:

[... *anusvāra*] + [*yay*...]
= [... nasal] + [*yay*...]

Thus by applying this rule we get forms like *kaṇtha, aìkita, guṃphita* etc. Consider however, the following example:

(5)	*kuṇḍam rathena*	
	kuṇḍaṃ rathena	... 8.3.23
	kuṇḍaṇ rathena	... 8.4.58

The resultant form here is not acceptable in Sanskrit. This is again overgeneration.

One may argue about redundancy being the feature of use of the *pratyāhāra*s in the metalangauge of P. However, the tradition has taken pains in creating a constraint to check such forms in the form of statements in this regard. Patañjali (Pa) in the context of the above example says:

rephoṣmaṇām savarṇā na santi [2] (the sounds *r* and the sibilants do not have any homogenious(nasal)).

There are at least some constraints in the form of statements of the later commentators to check the overgeneration as shown above. However, in the case of phonetic doubling mentioned below, we see hardly any constraint to check the overgeneration.

[2] *Vyākaraṇa Mahābhāṣya* of *Patañjali*, 2001, Vol.1, p 130.

Table 2. Environment for Phonetic doubling

1	**2**	**3 Consonant Reduplicated**	**4**	Rule of Panini
vowel	*r/h*	*yar*	—	8.4.46
—	vowel	*yar*	No vowel	8.4.47
vowel	*Yaṇ*	*may*	—	K & Pa on 8.4.47
vowel	*may*	*yaṇ*	—	As above
—	*Ś ar*	*khay*	—	As above
—	*khay*	*Ś ar*	—	As above

2.2 Phonetic Doubling

P in his A has dealt with the process of reduplication at three places; (i) 6.1.1-12[3], (ii) 8.1.1-15, (iii) 8.4.46-52. (i) deals with the reduplication of verbal roots in the forms of present as well as perfect tense and also in forming complex verbal roots such as desiderative and frequentative. In a nutshell, this reduplication applies to the *aṅga* in Paninian terminology. (ii) deals with the reduplication of the entire *pada*. The last section in the A mentioned above, deals with the reduplication of the consonants. The paninian tradition has augmented the existing set of rules laid down by P in this section, in the form of *Vārttika*s (mainly written by K) in this regard and the later tradition has interpreted certain statements of *Patañjali* in such a manner that the resultant forms can only be termed as over-generated ones. The later paninian tradition has done this exercise at many places and has come up with such overgenerated forms. Such extreme cases are presented in this paper and an attempt is made to study the approach of the Paninian system to handle this phenomenon.

(6) *putrādinī tvam asi pāpe*
(Oh! son-eater woman, shame on you!)
puttrādinī sarpiṇī
(she-snake is son-eater.)

In this case, t is seen reduplicated alongwith the change in the meaning. This case is noted by 8.4.48.

Environment for Phonetic doubling: In the same section, some other phonemes are also noted for their reduplicated occurrence. K and Pa have also noted down this tendency in some other phonemes. These phonemes are- same mentioned in fn 2. In the following table 2 they are referred to as *yar*, as used by P. In the table 2, these rules are explained with all details, namely environments-prior and posterior. In table 2, 1 , 2 , 4 refer to the environment for phonetic doubling. The order indicates the positions of these environments and the position of the phoneme reduplicated. The examples for these two rows are-

(7) *h**aryy**anubhavaḥ*
(h *a-r-y anubhavaḥ* > phonetic doubling of y)

[3] More recently, Kiparsky in a forthcoming article available on his webpage, has discussed it.

(8) (a)*rāmā**tt***
(*rām ā-t*-(no vowel) > phonetic doubling of t)

(b) *s**udhdh**yupāsyaḥ*
(*s-u-dh-y upasyah* > phonetic doubling of y).

K and Pa on the environment for phonetic doubling: While commenting upon 8.4.47, K notes- *dvirvacane yaṇo mayaḥ* . On this Pa has a two fold comment. He says-

dvirvacane yaṇo maya iti vaktavyam.
Kim udāharaṇam yadi yaṇa iti
pañcamī maya iti ṣaṣṭhī
ulkkā valmmikam ity udahāraṇam. Atha maya
iti pancamī yaṇa iti ṣaṣṭhī
dadhyyatra madhvvatrety udāharaṇam

This means- In the rules dealing with the process of phonetic doubling, the words yaṇo mayaḥ should be stated. What is the example ? If *yaṇaḥ* (*yan* is *y, v, r, l*) is taken to be ablative and *mayaḥ* (*may* is all stops except nasal palatal) is taken to be genitive, then the examples are –

(9) *u**lkk**ā / va**lmm**ikam*

and if *mayaḥ* is taken to be ablative and *yaṇaḥ* is taken to be genitive, then the examples are-

(10)*da**dhyy**atra / ma**dhvv**atra.*

Same argument is applied to another statement of K, namely *śaraḥ khayaḥ*[4] which provides us with the following examples-

(11) ***sthth**ālī / **sthth**ātā*

(12) *va**tss**aḥ / **kss**īram / a**pss**arāḥ*

This way of interpreting the statements of K on the rules of P becomes a peculiar feature of the system of paninian grammar. Later tradition of paninian grammar thus by interpreting statements of K and Pa and P have noted down forms which we here address as overgenerated forms.

We note that this feature is also noted by non-paninian systems such as *Kātantra*. A commentary on *Prakriyā-Kaumudī* namely *Prakāśa* notes that according to *Kātantra* school the phonetic doubling in a particular case will give rise to only 32 forms and not more.[5]

[4] This statement means that *khay* is reduplicated if it occurs after *ś*ar and *ś*ar is reduplicated if it occurs after *khay. śar* stands for all the sibilants except h and *khay* stands for all the voiceless stops.

[5] *PrakriyāKaumudī*, 2000, Vol.I, p 158.

Twice Occurrences of same consonant in Sanskrit. It is noteworthy to study the structure of the consonant cluster in Sanskrit. A list of such clusters is available in Coulson Michael, 2003, p 22-24. We concentrate on a cluster of two consonant of same phonetic value. In other words, we concentrate on the twice occurrence of the same consonant. In the following table a list of such consonant clusters is provided. Table 3 shows us that the consonants which can have twice occurrence without applying the rules of phonetic doubling are-

Table 3. Twice Occurrences of same consonant

Consonant	Position	Consonant	Position
k	(i)Final + initial of the next word (ii)Prefinal	n	(i) Final + initial of the next word (ii)Pre-final
g	Final + initial of the next word	ṅ	Final + initial of the next word
c	Final + initial of the next word	ṇ	Final + initial of the next word
j	Final + initial of the next word	ś	Final + initial of the next word
t	Final + initial of the next word	ṣ	Final + initial of the next word
d	Final + initial of the next word	s	Final + initial of the next word
p	Final + initial of the next word	m	Final + initial of the next word
b	Final + initial of the next word		

A careful glance at this table will point out that all these consonants fall in the domain of the application of the phonetic doubling rules mentioned above in Table 2. Therefore, if the rule for phonetic doubling is applied to these already existing two consonants, we get three same consonants occurring one after another. Such a form is noted to exist optionally by P in the case of consonants except nasals by the 8.4.65.

Generation of Phonetic doubling in later tradition: A 17^{th} century grammar text, *Vaiyākaraṇa-SiddhāntaKaumudī* (VSK) records following cases of phonetic doubling-

(13) *rāmātt rāmādd.*/ VSK 206, *dvitve rūpacatuṣṭayam.* (in the forms *rāmāt* and *rāmād*, after applying the rules of phonetic doubling we get 4 forms).

(14) *aidhidhvam* / VSK 2258, *dhaḍhayor vasya masya ca dvitvavikalpātṣoḍaṣarūpāṇi.* (by reduplicating *v* and *m* when immediately before *dha* and *ḍha* we get 16 forms.)

(15)*saṃskartā* / VSK 138, *anusvāravatām anusvārasyāpi dvitve dvādasa.* (after reduplicating the *anusvāra* in the forms already containing it, we get 12 forms).

(16) *gavāk* / VSK 443

Cases (15) and (16) deserve a special attention as they pose a problem.

2.2.1 Generation of Phonetic Doubling in the Forms of saṃskartā

(15) *saṃskartā* This word is formed in the following way[6]-

sam + kartā
sam + s- kartā … 6.1.134
sar + s-kartā … 8.3.5
saṃr + s-kartā … 8.3.2 / 8.3.4
saṃs + s-kartā … 8.3.15

Along with this form there is an optional form that is available in which in place of *ṃ* there occurs an *anusvāra*. In the following two tables, forms with *ṃ* and *anusvāra* are presented. In Table 4 and 5, we see phonetic doublings of *s* , *t* , *k* and more problematically of the *anusvāra*. This phonetic doubling of *anusvāra* is based on the argument of K that *ayogavāha*[7] s are to be included in the *pratyāhāra aṭ* as well as *śar* by the statement-

ayogavāhānam aṭṣuṇatvam śarṣu jaś tvaṣatve. What's the check in stopping the generation of phonetic doubling of *anusvāra* ?

2.2.2 Generation of Phonetic Doubling in the Forms of *Gavāk*

The *Paninian Dhātupāṭha* notes that the root *añc* is used in two senses viz. *gati* (to go) and *pūjana* (to worship/ respect). In the sense of one who goes to a cow, following derivations take place according to the rules of A-

In the sense of one who worships a cow, following derivation takes place according to the rules of A. In the above tables, the underlined forms are the forms of a noun derived from a verbal root. Note that the difference in the forms in these tables is a mere *n* which has brought about a sea of change in the meaning as well as the form itself. That is why P has noted them with all their variations. When we take these 6 forms as the base and start adding the *sup* terminations, we get following tables for these two tables. Tables 8 and 9 correspond to Tables 6 and 7 mentioned above. These are the forms in neuter gender. There are certain specific processes for neuter forms. That is why they are selected here. In these tables, in each slot, there are many optional forms shown. They result out of the optional application of the rules namely, 6.1.109, 6.1.122 and 6.1.123.

In table 8 onwards, there are 7 rows: one each for the case (*vibhakti*). The vocative is not considered here. Needless to say that the 3 columns are for the

[6] I have to turn to Devnagari fonts for these two case to stress the amount of problem.
[7] The term *ayogavāha* refers to *anusvāra, visarga, jihvāmulīya and upadhmānīya*, Vyākaraṇa Mahābhāṣya of Patañjali, 2001, Vol.1, p 132.

Table 4. Forms of *saṃskartā* with a first nasal vowel

Forms	Explanation
संस्कर्ता संस्स्कर्ता संस्स्स्कर्ता	Deletion of first *s* - comment of Pa-*samo vā lopam ity eke*[1]. Phonetic doubling of 1^{st} *s* by P.8.4.47
संस्क्कर्ता संस्स्क्कर्ता संस्स्स्क्कर्ता	Phonetic doubling of *k* – *śaraḥ khayaḥ* referred to in Table 2 above.
संस्कर्त्ता संस्स्कर्त्ता संस्स्स्कर्त्ता संस्क्कर्त्ता संस्स्क्कर्त्ता संस्स्स्क्कर्त्ता	Phonetic doubling of *t* in all the above 6 forms by *dvirvacane yaṇo mayaḥ*
संस्कर्त्त्ता संस्स्कर्त्त्ता संस्स्स्कर्त्त्ता संस्क्कर्त्त्ता संस्स्क्कर्त्त्ता संस्स्स्क्कर्त्त्ता	2^{nd} Phonetic doubling of *t* in the first 6 forms by *dvirvacane yaṇo mayaḥ*
संस्कर्ताँ संस्स्कर्ताँ संस्स्स्कर्ताँ संस्क्कर्ताँ संस्स्क्कर्ताँ संस्स्स्क्कर्ताँ संस्कर्त्ताँ संस्स्कर्त्ताँ संस्स्स्कर्त्ताँ संस्क्कर्त्ताँ संस्स्क्कर्त्ताँ संस्स्स्क्कर्त्ताँ संस्कर्त्त्ताँ संस्स्कर्त्त्ताँ संस्स्स्कर्त्त्ताँ संस्क्कर्त्त्ताँ संस्स्क्कर्त्त्ताँ संस्स्स्क्कर्त्त्ताँ	Nasalization of the final vowel by 8.4.57 in all the 12 forms mentioned above. In all, we have 24 forms in this row.

numbers, singular, dual and plural respectively. The numbers within the squares indicate the number of forms in that square. Thus there are 49 forms in this table.

Thus there are 69 forms in all in the table 9. The final square in table 9 has got 9 forms. The last 3 forms are a result of the application of the statement of K[8] according to which the 1^{st} class consonant is replaced by the 2^{nd} class consonant of the same class. Thus we see here *k* is replaced by *kh*. These are the

[8] *cayo dvitīyā śari pauṣkarasādeḥ* / on 8.4.48.

Table 5. Forms of *saṃskartā* with an *anusvāra*

<u>Forms</u>	<u>Explanation</u>	<u>Forms</u>	<u>Explanation</u>
संस्कर्ता संस्स्कर्ता संस्स्स्कर्ता	Deletion of first *s* - comment of Pa *samo vā lopam ity eke* . Phonetic doubling of 1st *s* by P.8.4.47	संस्कर्त्त्ता संस्स्कर्त्त्ता संस्स्स्कर्त्त्ता संस्क्कर्त्त्ता संस्स्क्कर्त्त्ता संस्स्स्क्कर्त्त्ता सं`स्कर्त्त्ता सं`स्स्कर्त्त्ता सं`स्स्स्कर्त्त्ता सं`स्क्कर्त्त्ता सं`स्स्क्कर्त्त्ता सं`स्स्स्क्कर्त्त्ता	2nd Phonetic doubling of *t* in the first 12 forms by *dvirvacane yaṇo mayaḥ*
संस्क्कर्ता संस्स्क्कर्ता संस्स्स्क्कर्ता	Phonetic doubling of *k* – *śaraḥ khayaḥ* referred to in Table 2 above	संस्कर्ताँ संस्स्कर्ताँ संस्स्स्कर्ताँ संस्क्कर्ताँ संस्स्क्कर्ताँ संस्स्स्क्कर्ताँ सं`स्कर्ताँ सं`स्स्कर्ताँ सं`स्स्स्कर्ताँ सं`स्क्कर्ताँ सं`स्स्क्कर्ताँ सं`स्स्स्क्कर्ताँ संस्कर्त्ताँ संस्स्कर्त्ताँ संस्स्स्कर्त्ताँ संस्क्कर्त्ताँ संस्स्क्कर्त्ताँ संस्स्स्क्कर्त्ताँ सं`स्कर्त्ताँ सं`स्स्कर्त्ताँ सं`स्स्स्कर्त्ताँ सं`स्क्कर्त्ताँ सं`स्स्क्कर्त्ताँ सं`स्स्स्क्कर्त्ताँ संस्कर्त्त्ताँ संस्स्कर्त्त्ताँ संस्स्स्कर्त्त्ताँ संस्क्कर्त्त्ताँ संस्स्क्कर्त्त्ताँ संस्स्स्क्कर्त्त्ताँ सं`स्कर्त्त्ताँ सं`स्स्कर्त्त्ताँ सं`स्स्स्कर्त्त्ताँ सं`स्क्कर्त्त्ताँ सं`स्स्क्कर्त्त्ताँ संस्स्स्क्कर्त्त्ता	Nasalization of the final vowel by 8.4.57 in all the 36 forms mentioned above. In all, we have 36 forms in this row.
सं`स्कर्ता सं`स्स्कर्ता सं`स्स्स्कर्ता सं`स्क्कर्ता सं`स्स्क्कर्ता सं`स्स्स्क्कर्ता	Phonetic doubling of *anusvāra* by 8.4.47 in the 6 forms mentioned in Row 1 & 2.		
संस्कर्त्ता संस्स्कर्त्ता संस्स्स्कर्त्ता संस्क्कर्त्ता संस्स्क्कर्त्ता संस्स्स्क्कर्त्ता सं`स्कर्त्ता सं`स्स्कर्त्ता संस्स्स्कर्त्ता सं`स्क्कर्त्ता सं`स्स्क्कर्त्ता सं`स्स्स्क्कर्त्ता	Phonetic doubling of *t* in all the above 12 forms by *dvirvacane yaṇo mayaḥ*		

forms, we can say on the authority of A and K, which are actually spoken by people. So far there is no problem. When we apply the rules of phonetic doubling of certain consonants to these above mentioned Table 8 and 9, we start facing a problem.

Table 6. Derivation of *Gavāk* (one who goes to a cow)

go + *añc*	... (in the sense of to go)
go + *añc* + *kvin*	... A. 3.2.59
go + *ac*	... A 6.4.24, A.6.1.67
goc / *go ac*	... A 6.1.123
goc / *gavāc* / *go ac*	... A.6.1.109,122,123,

Table 7. Derivation of *Gavāñc* (one who worships a cow)

go + *añc*	... (in the sense of to go)
go + *añc* + *kvin*	... A. 3.2.59
go + *añc*	... A 6.1.67
goñc / *gava añc*	... A 6.1.123
goñc / *gavāñc* / *go añc*	... A.6.1.109,122,123,

Effects of phonetic doubling on forms in Tables 8 and 9. In Table 10 and 11, the reduplicated forms of the forms mentioned in Table 8 and 9 respectively are presented.

In the table 10, we note that the phonetic doubling of k, g,$\tilde{n}$, y, m has increased the number of forms (which are indicated in each square). The reasoning for the phonetic doubling of k, g, $\tilde{n}$ is 8.4.47. The reasoning for the phonetic doubling of y and m is the same as mentioned in Table 5 namely, *dvirvacane yaṇo mayaḥ* . We also note that there is a phonetic doubling of even a *visarga* in certain forms. The reasoning for this phonetic doubling is same as mentioned after table 5, namely, inclusion of *visarga* in the *pratyāhāra yar*. Also there is nasalization which is marked by a sign on certain forms which has added those many forms.

We note that in this table the following consonants apart from the ones mentioned in Table 7 are reduplicated- *ṅ*, *ṣ*. The reasoning for phonetic doubling of *ṅ* is 8.4.47 and for *ṣ* is the one mentioned in Table 4. namely, *śaraḥ khayaḥ*. We also note that in some forms even the *visarga* is reduplicated like in the previous table. In both these tables we also note that in some forms three consonants are simultaneously reduplicated. We also see that nasalization is marked with the sign in some forms.

Thus if we compare the above tables statistically we come up with the following picture-

Unduplicated	Duplicated + Nasalized
49 (Table 8)	196 (Table 10)
69 (Table 9)	267 (Table 11)

If we are adopting the Paninian framework for generating forms by machine we will face similar problems if we apply the rules of phonetic doubling.

Table 8. Declension of *Gavāc*

	1	2	3
1	गोऽक् गोऽग् गोअक् गोअग् गवाक् गवाग् 6	गोची	गोऽञ्चि गोअञ्चि गवाञ्चि
2	गोऽक् गोऽग् गोअक् गोअग् गवाक् गवाग् 6	गोची	गोऽञ्चि गोअञ्चि गवाञ्चि
3	गोचा	गोऽग्भ्याम् गोअग्भ्याम् गवाग्भ्याम्	गोऽग्भिः गोअग्भिः गवाग्भिः
4	गोचे	गोऽग्भ्याम् गोअग्भ्याम् गवाग्भ्याम्	गोऽग्भ्यः गोअग्भ्यः गवाग्भ्यः
5	गोचः	गोऽग्भ्याम् गोअग्भ्याम् गवाग्भ्याम्	गोऽग्भ्यः गोअग्भ्यः गवाग्भ्यः)
6	गोचः	गोचोः	गोचाम्
7	गोचि	गोचोः	गोऽक्षु गोअक्षु गवाक्षु

Table 9. Declension of *Gavāñc*

	1	2	3
1	गोऽक् गोऽग् गोअक् गोअग् गवाक् गवाग् 6	गोची	गोऽञ्चि गोअञ्चि गवाञ्चि
2	गोऽक् गोऽग् गोअक् गोअग् गवाक् गवाग् 6	गोची	गोऽञ्चि गोअञ्चि गवाञ्चि
3	गोचा	गोऽग्भ्याम् गोअग्भ्याम् गवाग्भ्याम्	गोऽग्भिः गोअग्भिः गवाग्भिः
4	गोचे	गोऽग्भ्याम् गोअग्भ्याम् गवाग्भ्याम्	गोऽग्भ्यः गोअग्भ्यः गवाग्भ्यः
5	गोचः	गोऽग्भ्याम् गोअग्भ्याम् गवाग्भ्याम्	गोऽग्भ्यः गोअग्भ्यः गवाग्भ्यः)
6	गोचः	गोचोः	गोचाम्
7	गोचि	गोचोः	गोऽक्षु गोअक्षु गवाक्षु

3 Proposed Solution

This overgeneration of forms is caused by-

(i) Redundancy of the *pratyahara*.

(ii) Application of the rules of phonetic doubling mechanically.

(iii) Application of statements and interpretations of later Paninian commentators.

To solve this problem we propose the following -

If we are going to apply the rules of phonetic doubling we must make a rule that -

1. The *visarga* should never be reduplicated.
2. An *anusvara* should never be reduplicated.

Table 10. Phonetic doubling in the declension of *Gavāc*

	1	2	3
1	गोऽक्क् गोऽग्ग् गोअक्क् गोअग्ग् गवाक्क् गवाग्ग् 6	गोच्ची	गोऽञ्चि गोऽञ्ञ्चि गोऽञ्च्चि गोअञ्चि गोअञ्ञ्चि गोअञ्च्चि गवाञ्चि गवाञ्ञ्चि गवाञ्च्चि 9
2	गोऽक्क् गोऽग्ग् गोअक्क् गोअग्ग् गवाक्क् गवाग्ग् 6	गोची	गोऽञ्चि गोऽञ्ञ्चि गोऽञ्च्चि गोअञ्चि गोअञ्ञ्चि गोअञ्च्चि गवाञ्चि गवाञ्ञ्चि गवाञ्च्चि 9
3	गोच्चा	गोऽग्भ्याम् गोऽग्ग्भ्याम् गोऽग्भ्भ्याम् गोअग्भ्याम् गोअग्ग्भ्याम् गोअग्भ्भ्या गवाग्भ्याम् गवाग्ग्भ्याम् गवाग्भ्भ्याम् गोऽग्भ्य्याम् गोऽग्भ्याम्म् गोऽग्भ्य्याम गोअग्भ्य्याम् गोअग्भ्याम्म् गोअग्भ्य्याम गवाग्भ्य्याम् गवाग्भ्याम्म् गवाग्भ्य्याम गोऽग्ग्भ्याम्म् गोऽग्भ्य्याम्म् गोअग्ग्भ्याम्म् गोअग्भ्य्याम्म् गवाग्ग्भ्याम्म् गवाग्भ्य्याम्म् 24	गोऽग्भिः गोऽग्ग्भिः गोऽग्भिःः गोऽग्ग्भिःः गोअग्भिः गोअग्ग्भिः गोअग्भिःः गोअग्ग्भिःः गवाग्भिः गवाग्ग्भिः गवाग्भिःः गवाग्ग्भिःः 12
4	गोच्चे	Same as above 24	गोऽग्भ्यः गोऽग्ग्भ्यः गोऽग्भ्य्यः गोऽग्ग्भ्य्यः गोअग्भ्यः गोअग्ग्भ्यः गोअग्भ्य्यः गोअग्ग्भ्य्यः गवाग्भ्यः गवाग्ग्भ्यः गवाग्भ्य्यः गवाग्ग्भ्य्यः गोऽग्भ्यःः गोऽग्ग्भ्यःः गोऽग्भ्य्यःः गोऽग्ग्भ्य्यःः गोअग्भ्यःः गोअग्ग्भ्यःः गोअग्भ्य्यःः गोअग्ग्भ्य्यःः गवाग्भ्यःः गवाग्ग्भ्यःः गवाग्भ्य्यःः गवाग्ग्भ्य्यःः 24
5	गोच्चः	Same as above (24)	Same as above 24
6	गोच्चः	गोच्चोः	गोच्चाम्

3. The rule of phonetic doubling should not be applied more than once to one consonant .
4. The rule of phonetic doubling should not be applied to more than one consonant simultaniously.

Table 11. Phonetic doubling in the declension of *Gavāñc*

	गोऽञ्च् गोअञ्च् गवाञ्च् पूजार्थे द्वित्व	
1	गोऽङ्ङ् गोअङ्ङ् गवाङ्ङ्	गोऽञ्ची गोअञ्ची गवाञ्ची
2	गोऽङ्ङ् गोअङ्ङ् गवाङ्ङ्	गोऽञ्ची गोअञ्ची गवाञ्ची
3	गोऽञ्चौ गोऽञ्ञ्चा गोऽञ्ञ्चौ गोअञ्चौ गोअञ्ञ्चा गोअञ्ञ्चौ गवाञ्चौ गवाञ्ञ्चा गवाञ्ञ्चौ 9	गोऽङ्भ्याम् गोऽङ्ङ्भ्याम् गोऽङ्भ्य्याम् गोअङ्भ्याम् गोअङ्ङ्भ्याम् गोअङ्भ्य्याम् गवाङ्भ्याम् गवाङ्ङ्भ्याम् गवाङ्भ्य्याम्
4	गोऽञ्ञ्चे गोअञ्ञ्चे गवाञ्ञ्चे	गोऽङ्भ्याम्म् गोऽङ्ङ्भ्य्याम् गोऽङ्ङ्भ्य्याम्म् गोअङ्भ्याम्म् गोअङ्ङ्भ्य्याम् गोअङ्ङ्भ्य्याम्म् गवाङ्भ्याम्म् गवाङ्ङ्भ्य्याम् गवाङ्ङ्भ्य्याम्म्
5	गोऽञ्चः गोऽञ्ञ्चः गोअञ्चः गोअञ्ञ्चः गवाञ्चः गवाञ्ञ्चः 6	गोऽङ्ङ्भ्याम्म् गोऽङ्भ्य्याम्म् गोअङ्ङ्भ्याम्म् गोअङ्भ्य्याम्म् गवाङ्ङ्भ्याम्म् गवाङ्भ्य्याम्म् 24
6	same as above 6	गोऽञ्चोः गोऽञ्ञ्चोः गोऽञ्ञ्चोः गोअञ्चोः गोअञ्ञ्चोः गोअञ्ञ्चोः गवाञ्चोः गवाञ्ञ्चोः गवाञ्ञ्चोः 9
7	गोऽञ्चि गोऽञ्ञ्चि गोऽञ्ञ्चे गोअञ्चि गोअञ्ञ्चि गोअञ्ञ्चे गवाञ्चि गवाञ्ञ्चि गवाञ्ञ्चे 9	गोऽञ्चोः गोऽञ्ञ्चोः गोऽञ्ञ्चोः गोअञ्चोः गोअञ्ञ्चोः गोअञ्ञ्चोः गवाञ्चोः गवाञ्ञ्चोः गवाञ्ञ्चोः 9

In order to remove the redundency, we have to rely upon the statements of the later comentators and take note of their statements and modify the rule accordingly.

References

1. Abhyankar, K.V. (ed.): Vyākaraṇa-Mahābhāṣya of Patañjali with Marathi Translation (7 volumes) (1943); D.E.Society (1943) reprint, Sanskrit Vidya Parisamstha, Pune (2007)
2. Böhtlingk, O. (ed.): Pāṇini's Grammatik, 1st edn. Motilal Benarsidass, New Delhi (1998); Aṣṭādhyāyī edited and translated into German
3. Balshastri, P. (ed.): Vyākaraṇa-Mahābhāṣya of Patañjali, alongwith Pradīpa, Udyota and Śabdakaustubha, Pratibha Prakashan, New Delhi (2001); edited originally by Shri. GuruprasadShastri, reedited by Prof. Balshastri

4. Dixit, G.K. (ed.): Paribhāśenduśekhara of Nāgeśa, alongwith the commentary Sarvamaṅgalā, Varanasi edn. Sampurnananda Sanskrit University (1987)
5. Sitaramashastri (ed.): Bṛhacchabdenduśekhara of Nāgeśa (in 3 parts), 1st edn. Sampurnananda Sanskrit University, Varanasi (1996)
6. Caturveda, G., Bhaskar, P. (eds.): Vaiyākaraṇa-Siddhānta-Kaumudī, alongwith Bāla-Manoramā and Tattvabodhinī. Motilal Benarsidass (1987)
7. Mishra, P. M. (ed.): Prakriyākaumudī with Prakāśa (in 3 parts). Sampurnanada Sanskrit University, Varanasi (2000)
8. Dalai, B.K. (ed.): Phonology, in Studies in Sanskrit Linguistics, pp. 64–75. Bharatiya kala Prakashan, New Delhi (2007)
9. Cardona, G., Jain, D. (eds.): Indo-Aryan Languages. Number 11 in Routledge Language Family Series. New Fetter Lane, London EC4P4EE (2003)
10. Cardona, G.: Recent researches in Paninian studies, 2nd revised edn. Motilal Benarsidas, Delhi (2004)
11. Coulson, M.: Teach yourself Sanskrit. Revised by Richard Gombrich and James Benson, Hodder Headline Ltd. (2003) (1st published in 1976)
12. Kiparsky, P.: Panini as a variationist. Centre of Advanced Study in Sanskrit, University of Poona, in collaboration with MIT press, Cambridge (Massachusetts, U.S.A.) and London (England) (1980)
13. Kiparsky, P., Staal, J.: Syntactic and Semantic relations in Panini. Modern Studies in Sanskrit (1988)
14. Kiparsky, P.: Reduplication in Stratal OT (forthcoming), `http://www.stanford.edu/~kiparsky/#recent-Papers`
15. Prince, A., Smolensky, P.: Optimality Theory: Constraint Interaction in Generative Grammar. Blackwell Publishers (2004) Final version of the widely circulated Technical Report, Rutgers University Center for Cognitive Science and Computer Science Department, University of Colorado at Boulder (1993)

Abbreviations

A - *Aṣṭādhyāyī*
K - *Kātyāyana*
P - *Pāṇini*
Pa - *Patañjali*
V - *Vārttika*
VSK - *Vaiyākaraṇa-Siddhānta-Kaumudī*

Issues in Combinatorial Analysis of Vedic Verbal and Nominal Forms

Boris Oguibénine

Université Marc Bloch, Strasbourg, France
oguibeni@umb.u-strasbg.fr

Abstract. Algorithmically determining differential features of Ṛgvedic verbs and nouns presents a series of issues that brings a fresh perspective on Vedic grammatical structures.

Inflectional categories of the Vedic language are signaled within verbs and nouns by so-called cumulative morphs (sets of markers, each set carrying several grammatical meanings occurring predominantly at the end of word forms). To discover grammatical categories of word forms, they are to be analyzed from right to left and focusing on the parts of words carrying only grammatical information. An analysis of RV.1.11 shows minimal final parts of words signaling verbs as distinct from nouns.

When homonymous sound combinations occur in verbs and nouns, our technique is to proceed more to the left focusing on subsidiary signs (submorphs with no autonomous grammatical meaning) in combination with inflectional endings.

Two issues in Vedic philology are briefly discussed: transfer of stems in Vedic nouns, and Vedic accentual patterns.

Keywords: Rigveda, Vedic grammar, Vedic verbal forms, Vedic nominal forms, morphs, morphology, text analysis, motif index, mythology.

1 General Principles

Processing Vedic texts by computer implies having a clear purpose for which it is being done. Indeed, it was initially undertaken by me to improve and complete a project that aimed to elaborate an index of mythological motifs of the *Ṛgveda* (for details see Oguibénine 1981:117-132).

My first purpose, properly philological, was to define what could be called a mythological motif in the *Ṛgveda*. Having analyzed how frequently certain types of utterances appear in the *Ṛgveda*, I drew the conclusion that in the Vedic hymns, any utterance with two slots filled with a noun and a verb respectively is the linguistic implementation of a mythological motif[1]. This bipolar structure comprehensively reproduces the content of the minimal mythological information provided in the

[1] Note that this is a working definition of the mythological motif adopted in order to compile an index of mythological motifs of the *Ṛgveda*. For an explanation of this definition and a sample analysis of a hymn (RV 1.49) see Oguibénine 1981: 120-121 sqq.

G. Huet, A. Kulkarni, and P. Scharf (Eds.): Sanskrit CL 2007/2008, LNAI 5402, pp. 320–327, 2009.

hymns. This definition of the mythological motif is by no means universally valid, nor would it be appropriate for every kind of text.

To apply this concept of the mythological motif, the following two questions are to be asked when working with a corpus of mythological data:

(1) *Who is doing what?*

and its counterpart

(2) *What is being done by whom?*

It is understood that both questions involve mythological characters (or actors, or agents, according to varying terminology) and their actions. Answers to these questions take the form of expressions linking the arguments and the predicates (that is, agents, actions, and attributes expressing states, properties, etc.).

The next problem in terms of computer processing is to make a distinction between verbal and nominal forms, i.e., distinguishing two major lexical classes.

Obviously, to distinguish between *inflectional categories* signaled by word forms in the hymns, it is necessary to isolate the categories' characteristic features in changing words while disregarding provisionally lexical meaning. In other words, our analysis is guided by considerations of combinations and substitutions of characteristic markers of grammatical categories to establish whether a word form belongs to either nominal or verbal lexical class. Although at this stage the analysis is focused on these two categories only, all types of pronouns are also taken into account.

In the language of the hymns, the categories are signaled by cumulative morphs, i.e. by language signs that simultaneously carry several grammatical meanings (for the notion "cumulative morphs" see Mel'čuk 1982: 30-31).

In Vedic, cumulative morphs occur most frequently at the end of word forms. I.e., word forms in the hymns are analyzed from right to left to calculate the number of graphemes (or phonemes) necessary and sufficient to decide unambiguously whether a word form is a noun or verb, but our analysis is focused on the printed text of the Aufrecht edition, i.e., on a graphic representation of the *Ṛgveda.*

As a sample, RV.I.11 is analyzed below and further observations are made.

Text of RV.1.11:
1.011.01a índraṃ víśvā avīvṛdhan samudrávyacasaṃ gíraḥ |
1.011.01c rathī́tamaṃ rathī́nāṃ vā́jānāṃ sátpatim pátim ||
1.011.02a sakhyé ta indra vājíno mā́ bhema śavasas pate |
1.011.02c tvā́m abhí prá ṇonumo jétāram áparājitam ||
1.011.03a pūrvī́r índrasya rātáyo ná ví dasyanty ūtáyaḥ |
1.011.03c yádī vā́jasya gómata stotṛ́bhyo máṃhate maghám ||
1.011.04a purā́m bhindúr yúvā kavír ámitaujā ajāyata |
1.011.04c índro víśvasya kármaṇo dhartā́ vajrī́ puruṣṭutáḥ ||
1.011.05a tváṃ valásya gómató 'pāvar adrivo bílam |
1.011.05c tvā́ṃ devā́ ábibhyuṣas tujyámānāsa āviṣuḥ ||
1.011.06a távāháṃ śūra rātíbhiḥ práty āyaṃ síndhum āvádan |
1.011.06c úpātiṣṭhanta girvaṇo vidúṣ ṭe tásya kārávaḥ ||
1.011.07a māyā́bhir indra māyínaṃ tváṃ śúṣṇam ávātiraḥ |

1.011.07c vidúṣ ṭe tásya médhirās téṣāṃ śrávāṃsy út tira ||
1.011.08a índram ī́śānam ójasābhí stómā anūṣata |
1.011.08c sahásraṃ yásya ratáya utá vā sánti bhū́yasīḥ ||

Let us compare two series of cumulative morphs: (1) verbal endings and (2) nominal endings occurring in this hymn.

Verbal endings in their *sandhi* forms where applicable (verse numbers in parentheses):

-an (1), *-ma* (2b), *-mo* (2c), *-nty* (3b), *-te* (3d), *-ta* (4b), *-r* (5b), *-uḥ* (5d) , *-aṃ* (6b), *-´nta* (6c), *-uṣ* (6d) , *-ḥ* (7b), *-uṣ* (7c), *-a* (7d) , *-(a)ta* (8b), *-´nti* (8d)[2].

Nominal endings in their *sandhi* forms and with the final phonemes (graphemes), appearing before nominal endings, as, e.g., *-śv-* before *-ā* (*víśvā*, 1a) or *-ír-* before *-´aḥ* (*gíraḥ*, 1b); pronouns and their endings are included:

-´ram, *-´śvā* (1a), *-´asam*, *-íraḥ* (1b), *-mam*, ***-thī́nām*** (lc), *-´jānām*, *-´tim*, *-´tim* (1d); *-yé*, *#ta#*[3], *-jíno* (2a), *-vasas*, *-ate* (2b), #***tvā́m***# (2c); ***-vī́r***, *-´rasya*, *-táyo* (3a), *-tayaḥ* (3b), *-´jasya*, *-´mata* (3c), ***-tṛ́bhyo***, *-ghám* (3d); ***-rā́m***, *-dúr*, *-úvā*, *-´vir* (4a), *-´ujā* (4b), *-´ro*, *-´vasya*, *-´rmaṇo* (4c), ***-rtā́***, ***-rī́***, ***-utáḥ*** (4d); *#tvám#*, *-lásya*, *-´mató* (5a), *-ivo*, *-´lam* (5b), #***tvā́m***#, ***-vā́***, *-´uṣas* (5c), *-´āsa* (5d); *#ahám#*, *-ra*, *-íbhiḥ* (6a), *-´dhum* (6b), *-vaṇo* (6c), *#**ṭe**#*, *#tásya#*, *-rávaḥ* (6d), ***-ā́bhir***, *-ra*, *-ínam* (7a), *#tvám#*, *-´ṇam* (7b), *#**ṭe**#*, *#tásya#*, *-´rās* (7c), *-´ṣām*, *-āṃsy* (7d), *-´ram*, *-´nam*, *-´asā* (8a), *-´mā* (8b), *-´ram*, *#yásya#* (8c), *-táya* (8c), ***-´īḥ*** (8d).

In these two series, some sequences are identical. For example, the sequence *-´am* marks the AccSgmn in the nominal endings. But in the verbal form 6b (*práty*) *āyam*[4], it signals the 1SgImpf. Other examples are *-ta* (4b *ajāyata* 3SgImpf; 8b *anūṣata* 3PlAor and 3c *gómata* GenSgm) and *-ra* (7d *tira* 2SgImp*,* 6a *śūra* VocSgm). In order to distinguish them by automatic search, we have to consider the immediately preceding phonetic context, i.e. the units (graphemes) to the left.

Based on this evidence, a simple rule can be formulated:

(1) Any prepausal sequence *-´am* and *-ám* signals a nominal form.
(2) *-(a)ta* signals a verbal form, but *-´mata-* is a marker of a nominal form.
(3) *-(i)ra* signals a verbal form, but *-ūra* is a marker of a nominal form.

Among the verbal forms, a noteworthy example is the sequence *-nti/-nty*. Only three nouns with a prepausal *-nti* or *-nty* sequence (depending on a preconsonantal or prevocalic position) occur in the RV: *tánti* "rope", *ránti* "joy", *śakúnti* "bird"; the three nouns occur in forms other than their NSg in *-nti* or *-nty* (*tantáyas*[1]*; rántayas*[1]*, rántayo*[1]*; śakunte*[1] (Voc.), *śakúntayaḥ*[1])[5].

---

[2] *-´nti* : the stem is accented.
[3] The forms between # signs are either hapaxes (see below) or those in which stems and declensional endings cannot be separated (i.e., the suppletive pronoun stem).
[4] Cf. the non-preverbated form *ā́yam* 1SgImpf (RV. 1.125.3, 10.108.10).
[5] The superscript indicates the number of occurrences.

Another rule can thus be formulated:

(4) ***-nti* and *-nty* sequences signal only verbs**.

The accent is a graphically and phonetically efficacious means of distinguishing between the two categories - there are two unaccented verbal forms in RV.1.11 where the accent falls on preverbs: the *Padapāṭha* forms are *úpa atiṣṭhanta* (6 c), *áva atiraḥ* (7b), the exception being *sánti* (8d)[6].

Thus, to achieve the purpose of distinguishing the forms, it would suffice to give such rules as:

(5) ***-ma* signals a verbal form, if the form is unaccented** (see 2b *mā́ bhema*);

and similarly for

(6) ***-mo*, if occurring before a soft (voiced) consonant in the next word form** (2c *ṇonumo jétāram*),

etc. It would then be possible to eliminate wrong attributions of words to one of the two lexical classes.

But the difficulty mentioned above still remains: it is rather statistically normal that the last two phonemes of verbal and nominal endings are not a sufficient indication of the class. Examples are numerous: *-uḥ* as a verb ending (5d *āviṣuḥ*) appears as the ending of NSgm of the *u*-sterns; *-ma* presumed to signal a verb form characterizes NAccSgn of the *-man*-stems, etc.

I am aware that the term "category" is a generalization. But this is precisely the scope of the project being presented to make it explicit that both practically and linguistically it is quite useful to answer the following question: how long do final phonemic sequences of Vedic words have to be in order to determine the part of speech of the respective words?

This paper asserts that a new interpretation of the Vedic inflectional system different from its usual grammatical analysis should be based on exhaustive evidence that should include both standard inflectional units with as many preceding phonemes (graphic signs or graphemes) as necessary to distinguish between nominal and verbal forms occurring in the *Ṛgveda*.

2 On Some Related Philological Problems

We have seen that as far as the lexical classes are concerned, the procedures to differentiate them are based on final phonemic (or graphemic) strings. The length of these strings varies depending on inflectional endings and the minimal number of the preceding signs (graphemes/phonemes). Thus, a characteristic feature of our analysis is that it uses the minimal number of inflectional endings carrying grammatical information in combination with those parts of word forms that do not carry any grammatical information.

Here, we use a notion of the end of a word that is different from how it is defined in usual Vedic grammars. Our approach disregards the problem of morph boundaries and looks for relations between morphs that signal lexical classes and preceding

[6] On verbal accentuation see Klein1992.

phonemes (graphemes) that do not. Final sequences determined by this kind of analysis are more informative than grammatical morphs, because they include more phonemes.

In some cases, the parts of speech may be identified by morphs belonging to the Vedic grammatical system [e.g. of such morphs as *-sya* (InstrSg of the *a*-stems)]. It can be safely said that they are affixed only to nominal a-stems (3a *índrasya*, 3c *vā́jasya*, 5a *valásya*) and appear in pronominal adjectives and pronouns (4c *víśvasya,* 6d, 7c *tásya*, 8c *yásya*).

No verbal form has the final sequence *-asya*, while the verbal form #*asya*# (for example, RV.3.30.17, 6.52.3, etc.) is excluded on two grounds: (1) it is an independent morph (ImpvSg of *as*- "throw"), and (2) it is unaccented; be it noted that such cases as 3.24.1 *ápāsya* "throw it away!" or 3.50.1 *ā́sya* have to be considered separately because in the *Padapāṭha* they take the forms of *ápa asya* and *ā́ asya* respectively. Thus, searching for such forms would require a full list of all the preverbs.

It would be pertinent here to point out some special problems of Vedic as well as of general linguistics that can be studied in the future either in connection with or on the basis of our approach to the Vedic inflectional system.

1. Although our treatment of the inflectional system in the *Ṛgveda* has been elaborated independently, it appears that it basically corresponds to some ideas from the grammatical tradition of Ancient India. It is well known, for example, that Indian grammarians carefully distinguished between the *saṃhitā* and the *pada* shapes of textual evidence. This implies special attention given to the synthesis of utterances out of a finite number of the primary elements presented by Pāṇini in an abstract form. Thieme has stated that the term *vyākaraṇa* is to be understood as "an instrument by which forms are created in various ways" and the term *sūtra* as "means of sewing [together]" (Thieme 1982/1983)[7].

Thus, Pāṇini's usage of terms points in the direction that we have chosen. Both antecedent and subsequent positions of phonemes and grammatical elements play a prominent role not only in Pāṇini's system, but also in grammatical treatises like the Vedic *Prātiśākhyas*. Two precedents are equally worth mentioning, should there arise a rather legitimate question of to what extent our approach (nominal and verbal endings by themselves do not help to recognize the parts of speech automatically) is credible. These precedents stem from an Ancient Indian linguistic technique. On the one hand, Yāska's *Nirukta* classifies words as nouns, verbs, prefixes, and particles. The first two classes are established by definition, the remaining by enumeration (Scharfe 1977: 119). This is analogous to our approach: we record grammatical sequences (morphs) of the nominal system that are preceded by subgrammatical sequences, whereas the verbal system is characterized by either absence or different distribution of the same phonemic (graphemic) sequences that belong to the nominal system.

[7] Note that G. Cardona sees the term *vyākaraṇa* as an equivalent of the term *śabdānuśāsana* and states that "the object of the action denoted by *vy-ā-kṛ* is correct speech forms and "a grammar (*vyākaraṇa, śabdānuśāsana*) such as Pāṇini's is traditionally viewed as a means of explaining, making known" (see Cardona 1988: 665-666).

On the other hand, although it is true that the *Taittirīya Prātiśākhya* and the *Ṛgveda Prātiśākhya* tend to avoid grammatical expressions and instead provide a reasoning which is clearly at odds with the grammatical classification of nouns and verbs, the following remark of Scharfe's renders such a method plausible: "it is unbelievable that [the] author [of the *Taittirīya Prātiśākhya*] was ignorant of this ancient division and, therefore, his attitude reveals rather a sophisticated restraint than primitive clumsiness!" (Scharfe 1977: 128). It may be said that our approach while disregarding the firmly established traditional categorial distinctions views grammatical data on a par with subgrammatical data as a manifestation of the differentiating mechanisms in language that ultimately pursues its goals by substitutions of identifying units.

2. Thus, it is important to determine to what levels the identifying units belong. They cannot belong to the regular morphology level as neither verbal nor nominal morphology of Vedic traditionally operates with concomitance of morphs and phonetic environment that defines respective lexical classes and distinguishes between them.

Phonetic environment (called submorphs in some grammatical analyses) is actually not used in morphological research much, which is explained by the fact that linguists do not believe in interrelations between the smallest fragments of the grammatical signifiant and their signifié. However, as shown by Mel'čuk, Jakobson has successfully analyzed submorphs in his grammatical studies (for a technical definition of the submorph see Mel'čuk 1982: 108-109). Mel'čuk (1983: 65) adds that in a full-fledged description, the analyst must also take into consideration cases where a part of a morph, i.e. an individual phoneme or even a phonemic feature, becomes a meaning carrier. E.g., Jakobson observed that in Russian declension, the phoneme /m/ and its palatalized variant /m'/ are found in the endings of the three cases Dative, Instrumental and Locative and that neither phoneme occurs in the endings of the cases that he termed as nonmarginal in contradistinction to the three already mentioned[8]. Accordingly, Vedic sequences used here for defining word categories are composed of units that do not belong to either morph. They appear at a juncture of two morphs, i.e. they comprise morphs preceded by a unit belonging to another morph so that the morphs' boundaries are overlapped.

Within this interpretation, stem morphs followed by desinence morphs obviously take an unorthodox shape, since it is useful for us to list NSg of the Ṛgvedic stems in *-´an* (m.) as *-´jā* , *-´rdhā, -´jmā, -´bhvā, -´kṣā, -ñca, -´śvā*, *-´rṣā*, but also, to give an example of irregular inflection, as *#ū́dhar#, #ū́dhas#, #ū́dha#, #ū́dhaḥ#, #ū́dho#.*

Thus we do not say, as usually done, that the stems in *-´an* (m.) have *-´ā* in NSg. Because it is clear that no verbal form that has the above sequences occurs, isolating them appears to be the only economical way to indicate that the words *rā́jan, -´mūrdhan, párijman, dvibárhajman, víbhvan, bṛhádukṣan, páñca*(*n*), *mātaríśvan, vṛ̥́ṣan, and sahásraśīrṣan* (and compounds that have some of them as their second member) are nouns. We may conclude, if tentatively, that subgrammatical sequences as listed above eliminate any ambiguity, because the stem type (*-´an*) is taken into account along with as many predesinential phonemes as necessary to define the word

---

[8] Cf. Jakobson 1971:113: "In the system of language we discern two levels: the grammatical pattern of meaningful elements and the underlying phonemic pattern of mere discriminatory marks".

class, which would otherwise not be possible if we indicated that only the morph *-´ā* out of the *-´an*-stems signaled the NSgm: that is because not only are the vowels *-´a* and *-´ā* most frequent phonemes in Vedic, but also they most frequently occur in the final position in both categories and in too many forms.

3. The study of subgrammatical units is also of crucial importance to the analysis of the Vedic poetic language that, as has been shown by Saussure and some more recent authors, recurs to anagrammatic devices and creates meanings in the hymns. Taking again RV I.11, it can be observed that in this hymn dedicated to Indra the rate of the initial syllable of the god's name *ín-* is rather high: it occurs not only in the name *Índra*, but also in several nouns that fall into two categories: in the first category, the accented syllable *-ín-* appears within the noun stem (*bhíndur*, NSgm, *síndhum*, AccSgm); in the second category, the same syllable is the final suffix of the stem. It is part of the declensional termination in 1c *rathī́nām* GPIm, and precedes the case mark of GSgm in 2a *vājíno*. The usage of *-ín-* and *ī́n* corresponds to the overall semantic objectives of the hymn, the morphological constraints of Vedic noun declension, and stylistic endeavors of Vedic poets[9].

It can also be noted that the quoted words with the sequence *-in-* are nouns only. This is an additional mark of the nominal system as opposed to the verbal system: indeed no verbal form in this hymn has the sequence *in* either within the stem or within its inflectional (conjugated) form. See, for example, forms of the root *mi-* "to build" in RV.10.18.13d *minotu* 3SgImpv and RV.4.56.1 *minván* 3 PlInj – in both the syllable *in* overlaps the morpheme boundary as it belongs both to the root and to the verbal stem-building infix.

Thus, we can state that the grammatical oppositions that operate here are twofold: those that are between the nominal forms: GPlm *rathī́nām* vs. IPlm *rathíbhis* (the latter not appearing here), etc.; and those that are between the nominal forms with the Inlaut *-ín-* (*-ī́n-*)- (*rathī́nām, vājíno*, 7a *māyínam*) and no verbal forms at all with the sequence *in*.

Regarding the so-called transfer stems, it is currently said that a form like NSgm *anarvāṇas* "irresistible" should normally be *anarvā́.* Macdonell's explanation (Macdonell 1910: 211) is that "*anarvā́ṇas* may have started from the Acc *anarvā́ṇam* while the n. *anarvám* may have been due to the f. (*áditir*) *anarvā́* which appears like the f. of the *a*-declension".

These pseudo-historical conjectures are based only on analogy and cannot be proven; it is more convenient to consider that

(1) *anarvā́ṇa* is a *-vā́ṇa* "stem" with preceding *-r-* (thus a "stem" in *-rvā́ṇa*) NSgm *anarvā́ṇas,* AccSgm *anarvā́ṇam*;

(2) *anarvá* (Grassmann, s.v.) is an *-a*-stem that may, for the sake of our description, be considered a "stem" in *-arvá* (taking into account preceding phonemic/graphic units);

---

[9] Considerations of the style of Vedic hymns are given by Elizarenkova 1999: 651: she holds that the sequences *in* and *ra* are variously implemented in RV.3.40, a hymn to Indra, e.g. *índra, prá tira, candrā́sa, índavaḥ, vr̥trahan, giráḥ.* One could add that these sequences appear chiefly in nouns, the only exception being *tira* 2SgImp.

(3) *anarván* is a normal stem in -(*r*)*ván*: NSgm *anarvā́*, AccSgm *anarvā́ṇam*, AbSgm *anarvā́ṇas*, LocSgm *anarván*.

Such an explanation is confirmed in many other cases of the "transfer stems".

Although our approach may appear unorthodox, keen observations by L. Renou (1975:50 sqq.) should be mentioned: in his survey of consonant alternations in Sanskrit (in particular, the interchange between the gutturals *k, g, gh* and the palatals *c, j, h* - etymological **jh*) he draws attention to the constraints of the system ("en raison des appels du système") noting that the palatal consonants predominantly appear before verbal inflectional terminations, while the guttural consonants are used before the nominal inflectional terminations. Such observations are worthy of further exploration in the light of the analysis proposed above.

References

1. Aufrecht, T. (ed.): Die Hymnen des Rigveda, vol. I-II. Akademie-Verlag, Berlin (1955)
2. Cardona, G.: Pāṇini: His Work and Traditions, vol. 1. Background and Introduction. Motilal Banarsidass, Delhi (1988)
3. Elizarenkova, T.J.: Problemy izuchenija poeticheskogo jazyka Rigvedy v svete obshix idej R.O. Jakobsona. In: Jakobson, R. (ed.) Teksty, dokumenty, issledovanija, pp. 648–655. Nauka, Moskva (1999)
4. Jakobson, R.: The Phonemic and Grammatical Aspects of Language in Their Interrelations. In: Jakobson, R. (ed.) Selected Writings, vol. II, pp. 103–114. Mouton, The Hague (1971)
5. Klein, J.S.: On Verbal Accentuation in the Rigveda. American Oriental Society. New Haven, Connecticut (1992)
6. Macdonell, A.A.: Vedic Grammar. Trübner, Strassburg (1910)
7. Mel'čuk, I.A.: Towards a Language of Linguistics. Fink, München (1982)
8. Mel'čuk, I.A.: Studies of the Russian Language. In: Halle, M. (ed.) Roman Jakobson: What He Taught Us, Slavica, Columbus (1983)
9. Oguibénine, B.: Ansätze für ein Rigveda-Motivverzeichnis. Vorläufige Ergebnisse und Schwierigkeiten. Indo-Iranian Journal 23 (1981)
10. Renou, L.: Grammaire sanscrite. Adrien Maisonneuve, Paris (1975)
11. Scharfe, H.: Grammatical Literature, Harrassowitz, Wiesbaden. A History of Indian Literature V :2 (1977)
12. Thieme, P.: Meaning and Form of the 'Grammar' of Pāṇini. In: Studien zur Indologie und Iranistik, vol. 8/9 (1982/1983)

Verbal Roots in the Sanskrit Wordnet

Malhar Kulkarni and Pushpak Bhattacharyya

Indian Institute of Technology, Mumbai, India
malhar@iitb.ac.in,
pb@cse.iitb.ac.in
http://www.iitb.ac.in

Abstract. This paper aims to present a way of storing Sanskrit Verbal roots in a proposed Sanskrit WordNet. The synsets of verbal roots are proposed to be created using all the available dhātupāṭhas. While doing so, it is shown that various formal as well as semantic features of the verbal roots noted by Pāṇini should be taken into account and stored. This will serve the purpose of disambiguation. It is also shown that verbal roots that denote a different meaning when they occur with upasargas should be stored separately and linked to the synset of the changed meaning. This feature is peculiar to Sanskrit WordNet. Since, IIT Bombay has already developed Hindi as well as Marathi WordNets, information related to storing verbal roots in these two is also presented.

Keywords: Wordnet- Sanskrit, Hindi and Marathi, Dhātupāṭha, Synset, Semantic and Lexical Relations, Yaugika and Yogarūḍha words.

1 Introduction

Wordnets (WN) are accepted worldwide as useful lexical tools for Natural Language Processing (NLP). Projects for building WNs of different languages of the world have been going on for quite some time.[1] The scenario for Indian Languages is also encouraging. Indian Institute of Technology Bombay (IITB) has successfully created WNs for Hindi and Marathi.[2] There have been more than 100,000 hits of the sites for these resources.

The importance of developing a Sanskrit WN (SWN), in the context of Indian Languages (ILs) cannot be over-emphasised. Languages in India are broadly categorized into three families, one of which namely, Indo-European, has Sanskrit as a major language historically. Many modern Indian Languages like Hindi, Marathi, Bengali, Gujrathi, Panjabi, Oriya, etc. have a substantial number of borrowed Sanskrit words. Even the grammars of these languages have categories of words called *tadbhava* (generated from Sanskrit) and *tatsama* (similar to Sanskrit). A SWN, it follows, can logically provide a natural platform for integrating IL WNs. Several institutes and scholars have been trying to undertake the task of building a SWN with various strategies. Not much of substance, however,

[1] http://www.globalwordnet.org/gwa/wordnet_table.htm
[2] www.cfilt.iitb.ac.in

G. Huet, A. Kulkarni, and P. Scharf (Eds.): Sanskrit CL 2007/2008, LNAI 5402, pp. 328–338, 2009.

is visible on this front. The main issue regarding the structure of a SWN that comes up at the time of discussion is that while building the SWN, traditional knowledge bases (*śastric* knowledge) should be used, and one should not blindly follow structures of existing WNs which are based on western concepts.

It is this particular aspect that is aimed at studying in the present paper.

2 Main Aim of the Paper

This paper aims to present a way of storing Sanskrit Verbal roots in the proposed Sanskrit WordNet. The synsets of verbal roots are proposed to be created using all the available dhātupāṭhas. While doing so, it is shown that various formal as well as semantic features of the verbal roots noted by Pāṇini should be taken into account and stored. This will serve the purpose of disambiguation.

We believe that theories in the two traditional schools, namely, Vyākaraṇa and Navya-Nyāya can be effectively used for the construction of a SWN. It is indeed a matter of great privilege for us to have certain theories propounded by these schools as the base which may not be the case for other Indian Languages. We also believe that use of the Vaiśeṣika ontology as developed by Navya-Nyāya on the one hand and the kāraka theory as well as the semantic structure theory developed by the Vyākaraṇa school on the other hand would be very useful in this regard. Since the morphology of Sanskrit is very rich and since the syntax is said to be embedded in the morphology, there is a large influence of morphology on any of these theories. We cannot do away with morphological considerations while building a SWN.

In fact, some attempts have been made so far to propose schemes of Sanskrit WNs. Anupam proposed a Sanskrit WN which had only 22 synsets. S. Mohanty, K.P. Das Adhikary, P.K. Santi and G.P. Rout presented a structure of a proposed Sanskrit WN. This was a general structure of limited use. Although it recognized four types of words in Sanskrit, namely, *yaugika*, *yogaruḍha*, *ruḍha* and *yaugika ruḍha*, it focused entirely on nouns. It also suggested using Vaiśeṣika ontology which is well accepted. It did not, however, take into consideration verbal roots, which form the morphological core of the Sanskrit language being the derivational bases of a large number of Sanskrit nouns. It is believed that an effective use of verbal roots would lead to the major goal of a WN, namely word sense disambiguation for Sanskrit in particular, and for other Indian Languages in general.

We here propose the following:

1. A structure based on the verbal roots: We believe we are well supported here by the traditional school of Vyākaraṇa which says *sakalaśabdānāṃ dhātumūlatvāt* (*Parama-Laghu-Mañjūṣā*) (Since all words are derived from verbal roots). This same text classifies words into derivable (*yaugika*), conventional or underivable (*rūḍha*) etc. elsewhere. This shows that even if the above mentioned quotation is not considered as a finally accepted view, it can cetainly be applied to *yaugika* type of words.

2. Create synsets of verbal roots and not of verbal forms: this is for obvious reasons, the main among them being the large number of verbal forms which can be stored and used with the help of a morphological analyser. We have taken for example all the roots meaning *gati* (movement) from all the dhātupāṭhas. We note that there are more than 300 verbal roots in Sanskrit noted by all the dhātupāṭhas (a list attached). They all form members of the synsets of the meaning concept *gati/gamana*. We propose to have the following features mentioned in SWN:
   (i) Semantic Tree- This is useful in order to understand the semantic and syntactic structure of the verbal root as well as the nouns that are generated by it. It will be as illustrated in figure 1.

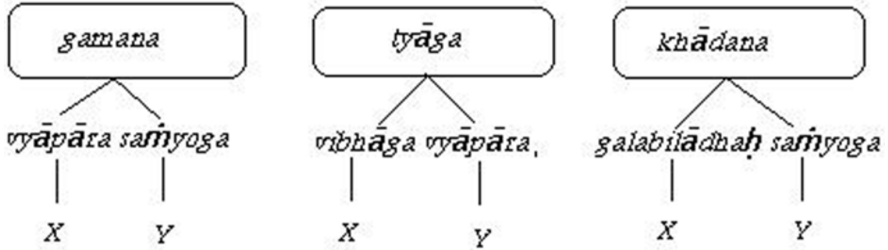


Fig. 1. Semantic and Syntactic Structure of Verbal Roots

   In this, x and y are not the same objects and the roots are called *sakarmaka*. Whereever these two are one and the same object, the roots are called *akarmaka*. This information is available to us from a semantic tree bank that will be developed for all the synsets of the verbal roots.

   (ii) Upasarga and meaning change- It is said that upasargas change the meaning of verbal roots.

   Upasargeṇa dhātvartho balād anyatra nīyate |
   prahārāhārasaṃhāravihāraparihāravat || (The meaning of a dhātu is perforce taken elsewhere by an upasarga; just as in the case of *hṛ*, when preceded by *pra*, means to strike, when preceded by *ā*, means to eat, when preceded by *sam*, means to kill, when preceded by *vi*, means to enjoy, and when preceded by *pari*, means to solve.)

We propose to link the original synsets of the verbal roots with the other synsets to which that root will logically belong after it is associated with a particular upasarga (see table 1). We would also like to store the following information regarding a verbal root (see figure 2).

1. Svara is useful for Morphological Analysis.
2. Gaṇa-vikaraṇa is useful in case of a root appearing in more than 2 groups with 2+ meanings.
3. Pada is helpful in cases like *bhuj*.

Table 1. Verbs and Upasargas

verbal sense	verbal roots	upasarga + verbal root	changed verbal sense	related verbal roots
gati	*gam*	*ava+gam* *adhi+gam*	*jñāna*	*jñā* *budh*
	hṛ	*sam+hṛ* *pra+hṛ*	*hanana*	*han* *hiṃs*

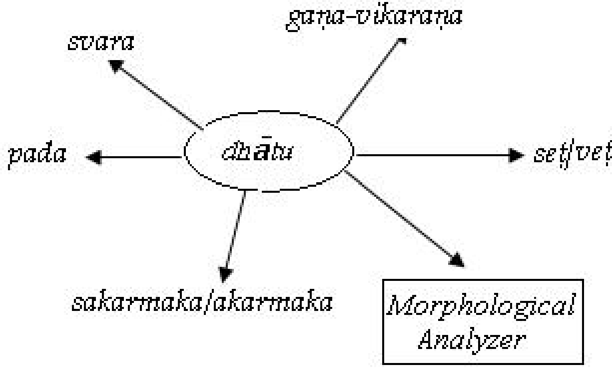


Fig. 2. Morphological and Other Information Stored with Verbal Roots

After having discussed various features of verbal roots that can be stored in Sanskrit Wordnet let us look at how Hindi and Marathi WNs are built and how verbal roots are treated therein. We have been engaged in building these two important lexical resources and think that the experience that we gained during the development of these two resources might help us in understanding the ways in which verbal roots in the Sanskrit Wordnet would be stored.

3 Hindi and Marathi Wordnets

We have, for long, been engaged in building lexical resources for Indian languages with focus on Hindi and Marathi (http://www.cfilt.iitb.ac.in). The Hindi and Marathi wordnets [2] and the HVKB [3] have been given special attention. The Wordnets more or less follow the design principle(s) of the Princeton Wordnet [1] for English paying particular attention to language specific phenomena (such as complex predicates) whenever they arise.

3.1 Hindi and Marathi Wordnets (HWN and MWN)

HWN and MWN have been created with the statistical features shown in table 2 compared with those of other wordnets:

Table 2. Current Status of Wordnets

	Total Number of Synsets	Total Unique Words
Hindi Wordnet	28,867	64,725
Marathi Wordnet	11,908	18,093
WordNet (2.1)	117597	155327
GermaNet (2004)	53312	76563
Multi Word Net (1.39)	32,700	58,000

Table 3. Details of Ontology

Part of speech	Number of nodes
Noun	151
Verb	39
Adjective	35
Adverb	14

HWN entry:
{peR, vriksh, paadap, drum, taru, viTap, ruuksh, ruukh, adhrip, taruvar} *'tree'*
jaR, tanaa, shaakhaa, tathaa pattiyo se yukt bahuvarshiya vanaspati
'perennial woody plant having root, stem, branches and leaves'
peR manushya ke lie bahut hi upayogii hai *'trees are useful to men'*

MWN entry:
{jhaaR, vriksh, taruvar, drum, taruu, paadap} *'tree'*
mule, khoR, phaanghaa, pane ityaadiinii yokt asaa vanaspativishesh
'perennial woody plant having root, stem, branches and leaves'
tii dammm jhaadacyaa saavlit baslii *'Being tired/exhausted she sat under the shadow of the tree'*

Fig. 3. MWN synset creation

We have incorporated a supporting ontology and have linked the synsets in the SWN to its nodes. The details of this supporting ontology are as follows: While HWN had been created from first principles by looking up the various listed meanings of words in different dictionaries, MWN has been created derivatively from HWN. That is, the synsets of HWN are adapted to MWN via addition or deletion of synonyms in the synset.

Figure 3 shows the creation of the synset for the word *peR* 'tree' in MWN via addition and deletion of synonyms from HWN. The synset in HWN for this word is {peR, vriksh, paadap, drum, taru, viTap, ruuksh, ruukh, adhrip, taruvar} 'tree'. MWN deletes {peR, viTap, ruuksh, ruukh, adhrip} and adds {jhaaR} to it. Thus, the synset for tree in MWN is {jhaaR, vriksh, taruvar, drum, taruu, paadap} 'tree'. Hindi and Marathi being close members of the same language family, many Hindi words have the same meaning in Marathi. This is especially so for tatsama words, which are directly borrowed from Sanskrit. The semantic relations are borrowed directly, thus saving time and effort.

Synsets. The principles of *minimality*, *coverage* and *replaceability* govern the creation of the synsets:

(i) Minimality: Only the minimal set that uniquely identifies the concept is used to create the sysnet, *e.g.*,
{ghar, kamaraa} (room)
ghar–which is ambiguous–is not by itself sufficient to denote the concept of a room. The addition of *kamaraa* to the synset brings out this unique sense.
(ii) Coverage: The synset should contain all the words denoting a concept. The words are listed in order of (decreasing) frequency of their occurrence in the corpus.
{ghar, kamaraa, kaksh} (room)
(iii) Replaceability: The words forming the synset should be mutually replaceable in a specific context. Two synonyms may mutually replace each other in a context C, if the substitution of the one for the other in C does not alter the meaning of the sentence. Consider,

{svadesh, ghar} (motherland)– {apanaa desh} (the country where one is born)
amerikaa meN do saal bitaane ke baad shyaam svadesh/ ghar lauTaa
America in two years stay after Shyam motherland returned
'Shyam returned to his motherland after spending two years in America'

The replaceability criterion is observed with respect to synonymy (semantic properties) and not with respect to the syntactic properties (such as subcategorization). For instance, the two verbs {aanaa, jaananaa} 'know' appear in the same synset for the word 'know'. In Figure 4, the sentence frames show that while *aanaa* 'know' assigns dative case to the *subject NP*, *jaananaa* 'know' assigns nominative case. The two verbs {aanaa, jaananaa} 'know' denote the same concept and each may replace the other in this particular semantic context. A synset in HWN (and in MWN) consists of the following elements.

A. Synset: {vidyaalay, paaThshaalaa, skuul} (school)
B. Gloss which consists of two parts.
a. **The text definition** that explains the concept denoted by the synset.
vah sthaan jahaaM praathamik yaa maadhyamik star kii aupachaarik shikshaa dii jaatii hai
'The place where formal education for primary or secondary level is given'

HWN entry:
{aanaa, jaananaa} *'know'*
kisiikaarya ko kame me samarthhonaa *'able to do something'*
Sentence Frames: NP1_DAT; NP2_NOM?: mujhe sillaii aati hai
1p,sg,DAT stiching know is
'I know to stich'
NP1_NOM; NP2_ACC: mai sillaii jaanti hai
1p,sg,NOM stiching know is

Fig. 4. Sentence Frame for 'know'

HWN entry:
{vidyaalay, paaThshaalaa, skuul} '*School*'
vah sthaan jahaan praathamik yaa maadhyamik star kii aupachaarik shikshaa dii jaatii hai
'*The place where the formal education of primary or secondary level is given*'
"is vidyaalay meM pahalii se paanchavii tak kii shikshaa dii jaatii hai"
'*education from first to fifth class is given in this school*'

MWN entry:
{shaaLaa, vidyaalay, paaThshaaLaa} '*school*'
jethe praathamik va maadhyamik staraavarii aupachaarik shikshaN dile jaate te ThikaaN
'*The place where the formal education of primary or secondary level is given*'
udyaa aamchyaa shaaLet vaktritvaspardhaa aahe
'*Tomorrow there is an elocution competition in our school*'

Fig. 5. HWN and MWN Sample Entry

Table 4. Criteria for Antonymy

Criterion	Examples	Gloss
Size	(chhoTaa-badzaa, moTaa -patlaa)	small-big, thick-thin
Quality	(achchhaa-buraa, pyaar-ghriNaa)	good-bad, love-hatred
Gender	(beTaa-beTii, maataa-pitaa)	son-daughter, mother-father
State	(shuruu-ant)	beginning-end
Personality	(raam-raavaN)	Rama-Ravana
Direction	(puurv-pashchim, aage-piichhe)	east-west, front-behind
Action	(lenaa-denaa, khariid-bikrii)	take-give, buy-sell
Amount	(kam-jyaadaa, halkaa-bhaarii)	little-much, light-heavy
Place	(duur-paas)	far-near
Time	(din-raat, subaha-shaam)	day-night,morning-evening

b. **A sample sentence** that uses the word in a sentence
is vidyaalay meM pahalii se paanchavii tak kii shikshaa dii jaatii hai
'Education from first to fifth class is given in this school'

The data is stored in the Devanāgari script in MYSQL database. The part of speech for each entry is listed in this database. In Figure 4 we provide sample entries from both HWN and MWN.

Lexical Relations. HWN incorporates commonly used semantic and lexical relationships along with a few new ones. A brief description follows:

1. **Antonymy** is a lexical relation indicating opposites. For instance, {**moTaa**, sthuulkaay} 'fat' → {**patlaa**, dublaa} 'thin'
patlaa (thin) is the antonym of moTaa (fat) and vice versa. The HWN also indicates the criterion under which the antonomy holds. In the above example, the antonymy criterion is *size*. Other criteria are given in Table 4.
2. **Gradation** is a lexical relation that represents possible intermediate states between two antonyms. Figure 6 shows the gradation relation among time words.

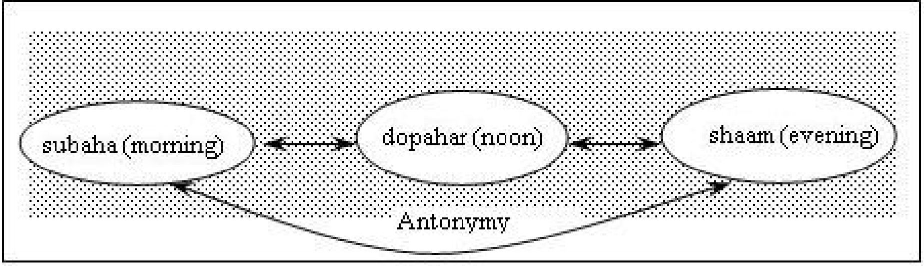


Fig. 6. Gradation relation

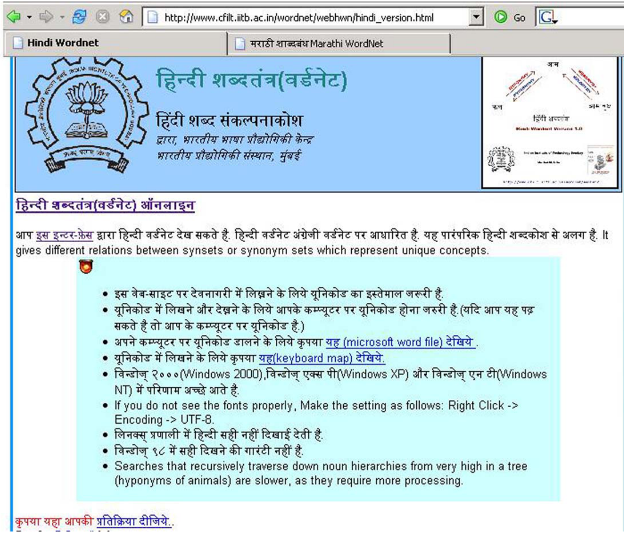


Fig. 7. Web Interface for Hindi Wordnet

3. **Hypernymy** and **Hyponymy** encode lexical relations between a more general term and specific instances of it.
   {belpatra, belpattii, bilvapatra} 'a leaf of a tree named bel
   → {pattaa, paat, parN, patra, dal} 'leaf'
   Here, belpatra (a leaf of a tree named *bel*) is a kind of pattaa (leaf). pattaa (leaf) is the hypernym of belpatra (a leaf of a tree named *bel*) and belpatra (a leaf of a tree named bel) is a hyponym of pattaa (leaf).
4. **Meronymy and Holonymy** express the *part-of relationship* and its inverse. {jaR, muul, sor} 'root' → {peR, vriksh, paadap, drum} 'tree' Here, jaR (root) is the part of peR (tree), implies jaR (root) is the meronym of peR (tree) and peR (tree) is the holonym of jaR (root).
5. **Entailment** is a semantic relationship between two verbs. The verb Y is entailed by X if by doing X you must be doing Y. For instance, *snoring* entails *sleeping*.

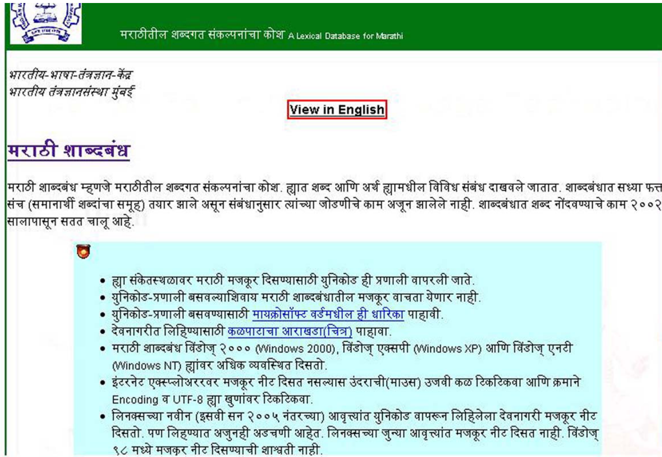


Fig. 8. Web Interface for Marathi Wordnet

{kharraaTaa lenaa, naak bajaanaa} *snore* → {sonaa} *sleep*

6. **Troponymy** is a semantic relation between two verbs when one is a specific "manner" elaboration of another. For instance,
   {dahaaRanaa} 'to roar' is the troponym of {bolanaa} 'to speak'
7. **Cross-linkage between different parts of speech:** The HWN also links synsets across different parts of speech. These links have not been taken from the EWN. Links between nouns and verbs include the following:

   (a) **Ability link** specifies the features inherited by a nominal concept. For example,
   {machlii, macchii, matsya, miin, maahii} 'fish' → {tairnaa, pairnaa, pauMrnaa} 'swim'
   (b) **Capability link** specifies features acquired by a nominal concept. For example,
   {vyakti, maanas} 'person' → {tairnaa, pairnaa, pauMrnaa} 'swim'
   (c) **Function link** specifies function(s) associated with a nominal concept. For example,
   {adhyaapak, shikshak} 'teacher' → {paRhanaa, shikshaa denaa} 'teach'

Links between nouns and adjectives are used to indicate typical properties of a noun. Example, {sher} 'tiger' →{maansaahaarii} 'carnivorous'. Links between morphologically derived forms mark the root form from which a particular word is derived by affixation. For example, {bhaaratiiyataa} 'Indianness' is derived from {bhaaratiiya} 'Indian' and is linked to it. Figures 1 and 8 below show the web interfaces for HWN and MWN, and Figure 3.1 shows the data-entry interface.

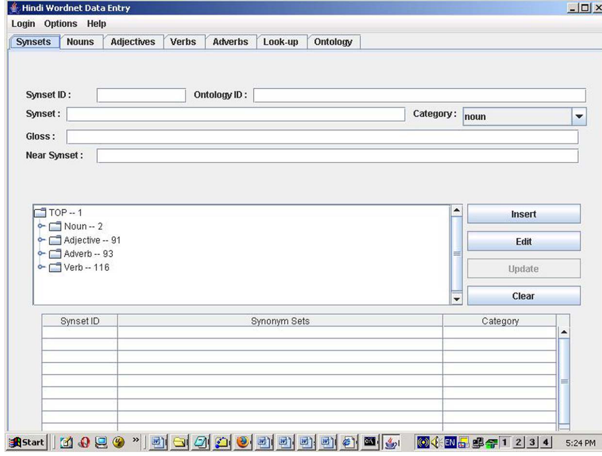


Fig. 9. HWN Data-entry Interface

4 Conclusion and Future Work

We propose to maintain the core structure of a WN as it is, while building the Sanksrit WN, in the sense that nodal elements will be synsets which will be linked with lexical and semantic relations. What we propose to add is the language specific approach which will include storing information related to morphology. This way of storing verbal roots will definitely cover almost all the *yaugika* words, as well as some of the *yogarudha* words.

Acknowledgements

We wish to thank Ms. Chaitali Dangarikar (PhD Student, Dept. of Humanities and Social Sciences, IIT Bombay) for invaluable technical support.

References

1. Anupam: Sanskrit as Indian networking language: A Sanskrit parser. Technical report, Indian Institute of Technology Kanpur, Kanpur (April 2004), `http://www.cse.iitk.ac.in/report-repository/2004/BTP_Anupam.pdf`
2. Bhagwat, V.B.: Paramalaghumañjuṣā with Marathi Translation. Dept. of Philosophy, University of Poona, Pune (2000)
3. Chakrabarti, P.D., Bhattacharyya: Creation of English and Hindi verb hierarchies and their application to Hindi wordnet building and English-Hindi MT. In: Proceedings of the Second Global Wordnet Conference, Brno, Czech Republic (2004)
4. Fellbaum, C. (ed.): WordNet: An Electronic Lexical Database. MIT Press, Cambridge (1998)

5. Mohanty, S., Adhikary, K.D., Santi, P., Rout, G.: Proposed model of Sanskrit word-net in concept capability of Sanskrit word-net: for convergence of knowledge-base. In: Convergence 2003 (2003)
6. Palsule, G.B.: The Sanskrit Dhatupathas: A Critical study. University of Poona, Pune (1961)
7. Palsule, G.B.: A concordance of Sanskrit Dhatupathas. Deccan College, Post Graduate Studies and Research Institute, Pune (1953)

An Effort to Develop a Tagged Lexical Resource for Sanskrit

S. Varakhedi[1], V. Jaddipal[2], and V. Sheeba[3]

[1] Sanskrit Academy, Hyderabad, India
shrivara@gmail.com
[2] Rastriya Sanskrit Vidyapeetha, Tirupati, India
v.jaddipal@gmail.com
[3] University of Hyderabad, Hyderabad, India
v.sheeba@gmail.com

Abstract. In this paper we present our efforts to design and develop a structured electronic lexical resource by tagging a Traditional Sanskrit dictionary. We narrate how the whole unstructured raw text of *Vācaspatyam* an encyclopedic type of Sanskrit Dictionary has been tagged to form a user friendly e-lexicon with structured and segregated information through corpus designing methods.

Keywords: Sanskrit, dictionaries, encyclopedia, thesaurus.

1 Introduction

It is not unknown to the scholars in the field of computational linguistics that electronic lexical resources are useful not only for human understanding but also for the needs of language processing. Many NLP applications like Morphological Analyzer etc. inevitably require a well-formed lexical resource. Lexical resource with grammatical and semantic information would be helpful in processing and translation and in decision making inference engines as well. It is not possible to do any kind of language processing in syntactic and semantic level without the structured relevant information regarding the stems of that language. Keeping this in view we have developed this e-resource for the Sanskrit language.

The Sanskrit language is one of the oldest classical languages of the world. It has a huge literary treasure related to all branches of sciences useful in all walks of life. Sanskrit is the first language to have a very precise grammar formalism authored by Pāṇini two thousand years ago. No other language has such a great tradition of grammar formalism, which is sound, perfect and very formal in nature. For these reasons Sanskrit gives enormous scope for NLP researchers and computer scientists to study this language from computational view point. Without proper lexical resources, NLP researchers will find themselves handicapped. This is the meeting point of traditional linguists and computational researchers. The information available in conventional lexicons is not sufficient for computational analysis. The way how information is stored becomes more crucial rather

G. Huet, A. Kulkarni, and P. Scharf (Eds.): Sanskrit CL 2007/2008, LNAI 5402, pp. 339–345, 2009.

than how much information is available in the lexicon. Therefore the lexicographer of an electronic lexicon should be careful while designing the lexicon for computational processing purpose. In case of Sanskrit the design of e-lexicon is more complex because the traditionally available lexicons or dictionaries in Sanskrit have many structural complexities. They are designed and organized for human understanding but not so well organized for computational processing purposes. Nevertheless they become important for they carry tremendous and immeasurable information. Hence restructuring of such available dictionaries in Sanskrit is the pre-eminent necessity of Sanskrit computational linguistics. In this direction, our team has opted *Vācaspatyam* for tagging on an experiment basis with a goal of developing a multipurpose electronic lexical resource that could be useful for academic research and computational processing as well. This work opens up further a new avenue of research in the development of Sanskrit electronic dictionaries following a similar method.

2 Hard-Book to Soft-Book

This encyclopedic type of lexicon was published in Kolkata in 1884. It is needless to say that the re-printed editions of this dictionary that are available now are not at all in readable condition due to old font types, unclear print and missing characters. Apart from all this, untraced errors in the original print often mislead the readers. It was felt necessary to have an electronic version of this gigantic work which runs into about five thousand pages in six big volumes printed in old kolkata printing halls using small Bengali style devanāgarī fonts. There are no breaks in words, not even clear breaks in topics and paragraphs. For each word entry tremendous information is collected. Nevertheless, everything is unorganised and hence not easily accessible even by eminent scholars. We started keying in the data into machines in ISCII using gist technology developed by CDAC Pune. Within a year we got the first raw e-version of the original text that needed several readings to get the proof corrected.

3 Introduction to the *Vācaspatyam*

Pandit Tārānātha Tarkavācaspati, with his in-depth erudition and indefatigable industriousness devoted several years to prepare thie encyclopedic Sanskrit Lexicon, consisting of about 5442 printed pages of A4 size. It contains terms along with their derivations and explanations drawn from almost all the branches of Sanskrit Literature, such as the Vedās, Vedāṅgās, Purāṇās, Upapurāṇās, Philosophy, Tantra, Arthaśāstra, Alaṅkāraśāstra, Chandaśāstra, Sangītaśāstra, Military Science, Pākavidyā, Sikṣā, Kalpa, Hasti Śāstram, HaṭhaYoga and Vāstuśāstra etc. Besides these, the technical words and doctrines of the following systems of Philosophy are fully explained: Cārvāka, Mādhyamika, Yogācāra, Vaibhāśika, Sautrāntika, ārhata, Rāmānuja, Mādhva, Pāśupata, Śaiva, Pratyabhijña, Raseśwara, Pāṇinīya Vyākaraṇa, Nyāya, Vaiśeṣika, Mīmāmsa, Sāṅkhya, Patañjali-YogaŚastra and Vedānta.

Sanskrit has many Lexicons such as Amarakośa, Vaijayanti, Viśwarūpakośa, Nānārthakośa etc., which serve more like thesaurus than a dictionary. However, the *Vācaspatyam* as a lexicon of Sanskrit arranges the words in alphabetical order and gives grammatical information with word-derivation as per the Pāṇinian System. Though the *Vācaspatyam* is constructed in the style of Śabdakalpadruma of Radhakantadeva, it excels Śabdakalpadruma in references and size.

This Sanskrit lexicon which is an encyclopedia in nature is a pioneering work of its kind and ever since its publication has been held in the highest esteem not only in India but even outside, because it is by far, wider and deeper in scope than any other contemporary Sanskrit dictionary.

4 Development of E-*Vācaspatyam*

The *Vācaspatyam* contains about 46970 unique word entries. Each entry has a minimum of 2 lines of information and in most of the cases it runs about 10 to 20 lines. Some prominent words may contain very elaborate content upto 20 pages on their category, meaning, sources, usages and other related information. More than 200 source books of different disciplines of learning are cited. References of more than 30 kośās (lexicons) are found. Names of these references and sources that were cited in short abbreviations are expanded to their full form for the benefit of the readers.

5 Tagging Scheme for E-*Vācaspatyam*

As soon as the basic electronic text was ready, we started to tag the text with meta-tags to identify the structures of the lexicon. The following tagset was developed for marking.

Tag	Description	Example
1. <cat>	Category of the word	a <cat> avyaya </cat>
2. <vp>	Vyutpatti i.e., etymology	aja <vp> na jāyate </vp>
3. <pr>	Prayogaḥ	usage in any standard Sanskrit work
4. <vkr>	Vyākaraṇa information	aja<vkr> na+janī+da </vkr>
5. <ar>	Artha i.e., meaning	aja <ar> caturmuKa </ar>
6. <vg>	Vigraha	i.e, explanation given for compound word
7. <akr>	Ākāra	i.e., source for the word or its etymology, usage etc.
8 <vn>	Vivaraṇam	i.e., narration about the particular concept

This tagset is used to segregate the semantic part of the lexicon. For stylistic presentation, we have used some other tags (<sl> to indicate Śloka etc.) which are not listed here. Initially the tagging was done manually with the help of some scholars. Later, we could find out some heuristics using which 70% of the text was mechanically tagged. Through this method we saved human labor.

The raw text added with these meta-tags, which contain necessary information about linguistic features, has become a good resource not only for presentation but also for better understanding of the original text and the semantics of the words listed in the lexicon.

6 Tree Structure

The following is the tree structure that took shape after tagging using XML.

– Simple structure 1

```
<word> %stem%
   [<cat><gr>|<gr></cat>]
       [[<m>[<gr>]?[pr]?[ref]?]+]+
</word>
```

```
+ indicates one or more occurances.
? indicates zero or 1 occurances.
```

Thus the lexicon that was readable only by an intelligent scholar, is made very simple in structure in order to be understood even by a novice in Sanskrit and also became usable for computational processing. This structure enabled us to develop a e-dictionary with different kinds of search options.

7 Complexities in Tagging

There were many instances of structural discrepacies in the text. We note some variations here such as

– Simple structure 1

```
<word>\\
     <cat>content</cat>\\
       [<m>content<m>]+\\
         [<ref>content</ref>]+\\
            [<ex>content</ex>]+\\
</word>\\
```

However it sometimes goes as follows.

– Simple structure 2

```
<word>\\
    [<cat>content</cat><gr>content</gr>]
         [<m>content<ref>content</ref>[<gr>]?[<eg>]?<m>]+
</word>\\
```

Further in some places the text has following structure.

– Simple structure 3

```
<word>\\
    <cat>content</cat><gr>
       [[<m>]+[<gr>]?[<ref>]?]+
</word>\\
```

Further, there are many instances where the meaning of root and suffix is described separately. However, sometimes either one of them or both may be missing, leading to the following structure within grammar information.

– Simple structure 4

```
<word>\\
    <gr>root[m]?[suff][m]?|[m]?[suff]</gr>
                        content
</word>
```

8 E-*Vācaspatyam* on CD ROM

The first version of Vācapspatyam CD ROM is ready for public release with 6 kinds of search options. All 42,000+ word entries are indexed alphabetically. By selecting any word in the word-index, one can access information related to the selected word. In the second option, the user can enter any string he wants to search by clicking on soft keyboard designed for Sanskrit alphabets. If the string is available in key word list, machine displays the relevant information about the string entered by the user. The third search option helps the user in searching all related words to a particular concept like wordnet. This option is unique as it helps the user in getting all the related words while composing poems etc. The Fourth search option is yet another unique experiment for Sanskrit Manuscript editors. In this option two entry boxes are given, where the user can specify the starting and ending letters of the word missing in the manuscripts. The machine brings all the possible words that begin and end with the specified letters. This option is found very much useful for the editors, while reading damaged manuscript with missing letters and words. Another search option is also given for the user to search for usages and expressions taken from various texts for a particular string or word. One can even search for all words derived from a root or a word with any particular suffix. In addition to all these, word-game enriches the CD with an added value.

We hope that users of this CD-ROM will find it useful for their research and other applications. We also hope that this becomes a model for tagging of any Sanskrit or other Indian language lexicons.

9 Technical Information

The CD-R presentation is developed using Visual Basic. The system tools that are available in VB are used. To avoid problems in font display, the textual output is shown in Netscape browser (Version 4.5) using DV-TT-yogesh font developed for ISCII text. This method is a tested one and works in all platforms from windows-98 to windows-XP and gets rid of broken font display problem while presenting the text through the browser. At the development stage Perl was extensively used for tagging manually annotated text and for removing errors.

10 Further Scope of Research

It is needless to establish that such a work of developing e-lexicon added with information meta-tags is of high importance in language processing. However, the traditional Sanskrit lexicons that are rich in content and poor in organization from computational aspects need to be restructured for computational purposes. This work poses several challenges for lexicographic science in computational linguistics. There are dozens of such complicated dictionaries like Śabdakalpadruma and V.S.Apte dictionary for Sanskrit-English and vice-versa. Our team has started working on both of these dictionaries. We hope we will come with good results very soon.

Acknowledgment

Authors thank K V RamKrishnamacharyulu, Vineeth Chaitanya, Amba P Kulkarni, Deeptha and Anilkumar for useful suggestions at various stages of the work.

References

1. Tāranatha Tarka Vācaspati,Vācaspatyam (Reprint). Rashtriya Sanskrit Sansthan, New Delhi (2000)
2. Descartes, Bunce: Programing the Perl DBI. O'Reilly, Sebastopol (2000)
3. Ray, E.T., McIntosh, J.: XML and PERL. O'Reilly, Sebastopol (2002)
4. Friedl, J.E.F.: Mastering Regular Expressions. O'Reilly, Sebastopol (2002)

11 Appendix I: Sample Snapshot of Tagged Text

%नलद% <cat>न</> <vkr><vg>नलम् तृषाबन्धं द्यति</vg> दो-खण्डने क। </vkr> <ar>(1) उशीरे (वेणारमूल) <vn>उशीरशब्दे 1376 पृo तद्गुणा दृश्याः </vn></ar> <ar>(2)पुष्परसे</ar> <ar> (3)जटामांस्याञ्च मेदिo <pr>"करिणां मुदे सनलदाऽनलदा" किराo</pr> <pr>"नलदेनानुलिम्पन्ति" "नलदमालां प्रतिमुञ्चन्ति" आश्वoश्रौ o6 ।10 ।3 ।4 ।</pr> <vkr><vg>तत्पण्यमस्य</vg> किशराo ष्टन्। </vkr></ar> <ar> (4)नलदिक तद्विक्रेतरि त्रिo <vkr>स्त्रियाम् ङीष् ।<vkr><vg>नलं ददाति</vg> दा - क।</vkr></ar> <ar> (5) नलदातरि त्रिo <pr>"स्यादस्यानलदं विना न दलने तापस्य कोऽपि क्षमः" नैषधo ।</pr></ar> <ar> (6)रुद्राश्वनृपस्य घृताच्यां जाते कन्याभेदे स्त्री<pr>"खलदा चैव राजेन्द्र नलदा सुरसाऽपि च" हरिवंo31अo तत्कन्योक्तौ ।</pr></ar>
</न्ल्द्>

12 Appendix II: Sample Snapshots



Fig. 1. Books: Hard Copy



Fig. 2. Mainpage



Fig. 3. Editors help



Fig. 4. Synonyms

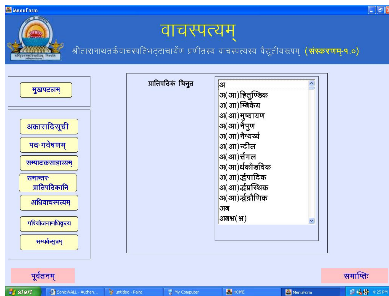


Fig. 5. word-searching



Fig. 6. Search

Towards a Scholarly Editing System for the Next Decades

Peter Robinson*

Institute for Textual Scholarship and Electronic Editing,
College of Arts and Law, University of Birmingham
Edgabaston, United Kingdom
p.m.robinson@bham.ac.uk
http://itsee.bham.ac.uk

Abstract. As almost all texts of interest to scholarly editors exist in multiple versions, efficient systems of collation are crucial. The first part of the paper explains desiderata for computer-based collation. The second part of the paper argues that the most radical impact of the digital revolution is to transform scholarly editing from the work of single scholars, working on their own on single editions, to a collaborative, dynamic and fluid enterprise spanning many scholars and many materials.

Keywords: Collation, scholarly editing, collaboration, digital methods.

1 Introduction: A Short History

As so often happens when one is asked to give a talk, and then to write it into a paper: the talk you thought you wanted to give when you were asked, isn't quite the talk that you want to give when you come to the moment, and when you come to write it into a paper, it changes again. When I was first asked to come to the Brown symposium, and was told what it was about, my head was full of collation, and so the original topic of the talk was "Towards a scholarly collation system for the next decades." This is actually quite an interesting thing to have in your head, as I hope this paper shows. However, in the months after I suggested this topic in early 2008, I became increasingly interested in the environment in which a collation system must exist. However, because scholarly collation is critical to any scholarly editing system, if it is to be of any use, scholarly collation will still figure largely.

First, let me establish my credentials, as some-one who might know some things about using computers to help scholars create useful collations of variant sources. I have been working on computer-assisted collation systems for over twenty years.

* This paper is based on the talk given at the Second International Sanskrit Computational Linguistics Symposium held at Providence, Rhode Island in May 2008. I am grateful to Peter Scharf and the Symposium Organizers for the opportunity to attend the symposium, and to the National Science Foundation which supported the symposium.

G. Huet, A. Kulkarni, and P. Scharf (Eds.): Sanskrit CL 2007/2008, LNAI 5402, pp. 346–357, 2009.

It all started in 1985 when Ursula Dronke, then Vigfusson Reader in Old Icelandic at the University of Oxford, suggested that I might edit the Old Norse poetic narrative sequence Svipdagsmal: there were, she said, a few manuscripts about and it might be interesting. There turned out to be 44 manuscripts: very interesting indeed. Right then, I bought my first computer, purely for word-processing the thesis. However this computer – an Amstrad PCW – also had a Basic interpreter and I found myself fascinated with programming it. So I attended Susan Hockey's courses on computer programming with SNOBOL for the humanities at OUCS, along the way also making the acquaintance of Lou Burnard. I had already started making electronic transcripts of the manuscripts, initially to make concordances. Now I developed a dangerous ambition: to write a computer program to compare the manuscripts. This became what you might call Collate 0: some 1200 lines of SNOBOL, later SPITBOL, code. For those who think intelligent life began with the iPOD, the SNOBOL family were pattern-matching programs specifically designed to process text: rather like PERL, I think.

Collate 0 created an efficient collation of my Icelandic manuscripts. Even better, I think, I was able to translate the whole collation output into a relational database and use database tools to help me work out the relations between the manuscripts. I described this work in two articles published in Literary and Linguistic Computing in 1989-90 on the collation and analysis of Icelandic manuscripts [1]. Following on from this, in 1989 Susan Hockey and I won a grant to develop my rather crude collation program, from something which could only be used by me, on materials which were just-so, only in the basement of the Oxford University Computing Services building between 3 and 4 am in the morning, to a program which could be used by any scholar, on any text, anywhere, any time.

So, in 1990, I began writing what became Collate 2 [2]. This was designed for the Macintosh computer, then running the state-of-the-art System 7. Collate 2 is still, eighteen years later, in continuous use in at least eight major editing projects. It still runs in Classic mode, only, and I have stopped answering questions on when there will be a Windows version. However, now that Apple has stopped all support of Classic, we have to hoard our old computers so that we can carry on running Collate.

Here is a list, very far from comprehensive, of who is using, or has used, Collate: the asterisk indicates that the scholar is still using collate.

Prue Shaw: Dante's Monarchia [3]
Prue Shaw and others: Dante's Commedia*
INTF-Münster/ITSEE Birmingham: Greek and Latin New Testament* [5][4]
Michael Stone: Armenian texts
Sid Reid: Conrad texts*
Eric Sneddon: Old French texts
Godfried Croenen: Old French Chronicles*
Tommy Wasserman: Gospel of Jude [6]
Wendy Phillips-Rodriguez: Sanskrit*
The author, Barbara Bordalejo, and others: the Canterbury Tales* [7]
Dorothy Severin/Fiona Maguire: the Spanish Cancioneros* [8]

John Kilcullen: William of Ockham [9]
Michael Bakker and others: Old Church Slavonic texts

I think this shows, rather nicely, that Collate achieved at least part of our aim: that it became one of that rather select group of humanities computing programs actually used by someone other than the person who wrote it. Indeed, even more rarely: it is used by people I have never met.

2 What Scholarly Collation Programs Must Do

After all this time spent making collation programs, what have I learnt? We have learnt two principles which I think must be fundamental to any good collation program. Collate is built on these two principles, and hence its success. We have learnt one thing it does not do, which a scholarly collation program should do. And we have learnt that in a crucial respect the design of Collate was fundamentally flawed and we must do better.

The first of the two principles on which Collate is built was rather nicely put by a student of mine, who said it this way:

Scholarly collation is not Diff.

There are, of course, an immense number of text comparison programs out there, most of them with the venerable Diff algorithm lurking somewhere deep within them. Figure one is the result of one of these programs, working on variant texts of a single line.

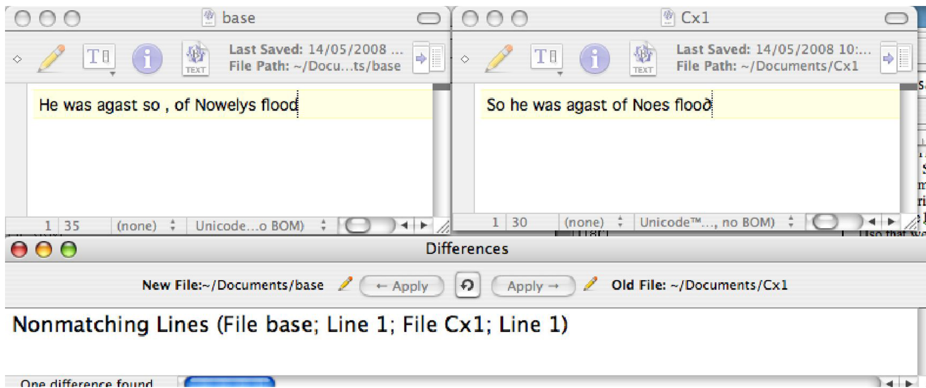


Fig. 1. The results of a 'Diff' comparison of one line of text in two sources

All it does is tell us that there is a difference between these two lines. Well, thank you very much. Now, there are more sophisticated programs which will identify differences at the word level. And it is very tempting to think: all we need do is find one of these, tweak it a bit, and we will get the perfect collation. I can not say this strongly or loudly enough. Anyone who thinks that there will ever, ever be a collation program, a kind of superDiff, which will give scholars, for any text at all, exactly the collation they want is naïve in the extreme. This

does not stop people trying to write such a program and, even, funders giving them money to do this. But there never, ever, will be such a program.

There are two reasons for this. First, any kind of automated program can identify one thing, and one thing only: it can identify differences. Any scholar will tell you that differences are not variants. Depending on just how you are reading a text: some differences will be just, well, noise: only a few, perhaps a very few, are real variants, of real interest to real scholars. We scholars spend years looking at differences, filtering them according to our interests at the time, developing a sense of what is significant and what is not. The notion that somehow we can build this knowledge into some kind of reductive process is absurd.

The second reason is that even if you could, by some miracle, teach a machine exactly what is significant, what is not, you would still have to teach it to work out exactly what is the best way of presenting any given sequence of variants, at any particular moment. Let's take our sample two lines, to show what a scholarly collator has to do:

He was agast so , of Nowelys flood
So he was agast of Noes flooδ

Now, our first judgement is going to be: I think the comma in the first line, and the non-standard d on 'flood' in the second line are just 'noise'. So now I have these two lines to compare:

He was agast so of Nowelys flood
So he was agast of Noes flood

But, exactly how should I present this? I could present it as a single long phrase variant:

He was agast so of Nowelys] So he was agast of Noes

Or I could present it as three variants, as a sequence of addition/omission/ replacement, thus:

He was] So he added
so] omitted
Nowelys] Noes

Or again as three variants, varying this, thus:

He was] So he
so] omitted
Nowelys] Noes

Or, as two variants:

He was agast so] So he was agast
Nowelys] Noes

For me, the latter is much the best. Now, I could get the computer to generate any and all of the alternatives. But what I could never, ever, get it to do is decide which alternative, in every context, is the best.

When I think back on the decisions I took when designing Collate, the one decision which I think most absolutely right then, and right now, is this: that the scholar should at every point have the ability to intervene in the collation and fix the variation just the way he or she wants it. That is: the scholar can say: this spelling here is not a real variant, for example the two forms of 'flood' in this example. And the scholar can say: at this point, I want the variation shown exactly so, for example as a sequence of two variants in this line, not one or three. These two functions correspond to the 'regularization' and 'set variants' routines within Collate.

The one reason above all that Collate has achieved the success it has is because it has this faciltity: it allows the scholar to fix the collation exactly as he or she wants. This is why it is used by the New Testament group in Münster, probably the most demanding collators anywhere. It still seems to me that this need is the most fundamental requirement of any scholarly collation program. Any program that does not offer this facility is just a dressed-up Diff program, a nice toy perhaps, but not a serious tool for serious use.

The second of the two principles on which Collate is built is this:

Collation is more than visualisation.

For most collation systems, the aim is to show the variation. Sometimes, they manage to show the variation in a rather beautiful form. But the problem is that they show it in one beautiful form only [1]. Again, one thing I did right with Collate, right back in the very beginning, was that I did not design the program so it could produce just one output. I designed it so that you could generate what you might call an intelligent output. Essentially, this distinguished all the different components of an apparatus - the lemma, the variant, the witness sigil, and much more – and allowed you to specify what might appear before and after each component, and in what order the various components might appear. This gave exceptional flexibility to the Collate output. You could use Collate to generate complex print editions, using (for example) the EDMAC macros developed by Dominik Wujastyk and John Lavagnino; or you could turn out the apparatus in HTML, SGML or (lately) XML [11]. This allows you to make complex electronic editions, such as that in Figure 2.

This figure shows the collation for the fifty-four manuscripts of the first words of Geoffrey Chaucer's Miller's Tale [12]: beside each word and variant is a list of the manuscripts containing that word and variant. Notice that at this level, all the information about spelling variation is filtered out. You can see that spelling variation also, as shown in Figure 3.

Alternatively, you could output the apparatus in a form ready for processing by an analysis program, to help you discover the relations among the witnesses. We have had great success in using evolutionary biology software to generate views of the relations between witnesses, as shown in Figure 4, for Dante's *Monarchia* [3].

I believe that a strong reason for Collate's success has been this flexibility of output. Put this together: the first thing Collate does well is to allow scholars to

[1] For example, the excellent JUXTA program, developed in the University of Virginia [10]

VMap	Whilom	Ad1 Ad2 Ad3 Bo1 Bw Ch Cn Cp Cx1 Cx2 Dd Dl Ds1 El En1 En3 Fi Gg Gl Ha3 Ha4 Ha5 He Hg Hk Ht Ii La Lc Ld1 Ld2 Ln Ma Mg Mm Ne Nl Ox1 Ph2 Pn Ps Pw Py Ra3 Ry1 Ry2 Se Sl1 Sl2 Tc1 Tc2 To1 Wy
	SOm tyme	Ra1
VMap	ther	Ad1 Ad2 Ad3 Bo1 Bw Ch Cn Cp Cx1 Cx2 Dd Dl Ds1 El En1 En3 Fi Gg Gl Ha3 Ha4 Ha5 He Hg Ht Ii La Lc Ld1 Ld2 Ln Ma Mg Mm Ne Nl Ox1 Ph2 Pn Ps Py Ra1 Ra3 Ry1 Ry2 Se Sl1 Sl2 Tc1 Tc2 To1 Wy
	þet	Pw
	▯	Hk
VMap	was	Ad1 Ad2 Ad3 Bo1 Bw Ch Cn Cp Cx1 Cx2 Dd Dl Ds1 El En1 En3 Fi Gg Gl Ha3 Ha4 Ha5 Hg Hk Ht Ii Lc Ld1 Ld2 Ln Ma Mg Mm Ne Nl Ph2 Pn Ps Pw Py Ra1 Ra3 Ry1 Ry2 Se Sl1 Sl2 Tc1 Tc2 To1 Wy
	was ~~þ²e was~~	La
	▯	He Ox1

Fig. 2. The collation for fifty-four manuscripts of the first words of Geoffrey Chaucer's Miller's Tale

VMap	Whilom	Sl2 (WHilom), Ds1 Ra3 (Whilom), Gg (WHilhō), Ad1 Gl La Ne (WHilom), Hk (WHilome), Sl1 Wy (WHylom), Ry1 (WHylome), Ad2 Ad3 Ch Cp Dd El En1 Ha3 Ha4 Hg Ht Lc Ld1 Mm Nl Pw Ry2 Se (Whilom), Tc1 (Whilom̄), Ii Ln Tc2 (Whilome), Dl (Whilum), Ox1 (Whylom), Bw (Wylom), Ps (Uhilom), Ph2 (W<u>hilom̄</u>), Cx1 Py (WHilom), Cx2 (WHylom), He (WWhilom̄), Ld2 Mg (Whilom), Cn To1 (Whilom̄), Pn (WHilom), Fi (WHilom̄), En3 Ha5 Ma (Whilom), Bo1 (w<u>Hilom</u>)
	SOm tyme	Ra1 (SOm tyme)

Fig. 3. The collation for fifty-four manuscripts of the first words of Geoffrey Chaucer's Miller's Tale

say: THIS is what the variation is. The second thing Collate does well is to allow scholars to say: I want the collation output in exactly this form. This is a very powerful combination, and any successor program that hopes to be as useful as Collate must include these capacities.

I said that we now know there is one thing Collate does not do, which we now think a collation program should do. Collate is extremely good at identifying variants of single words. It is very good in the basic identification of phrase

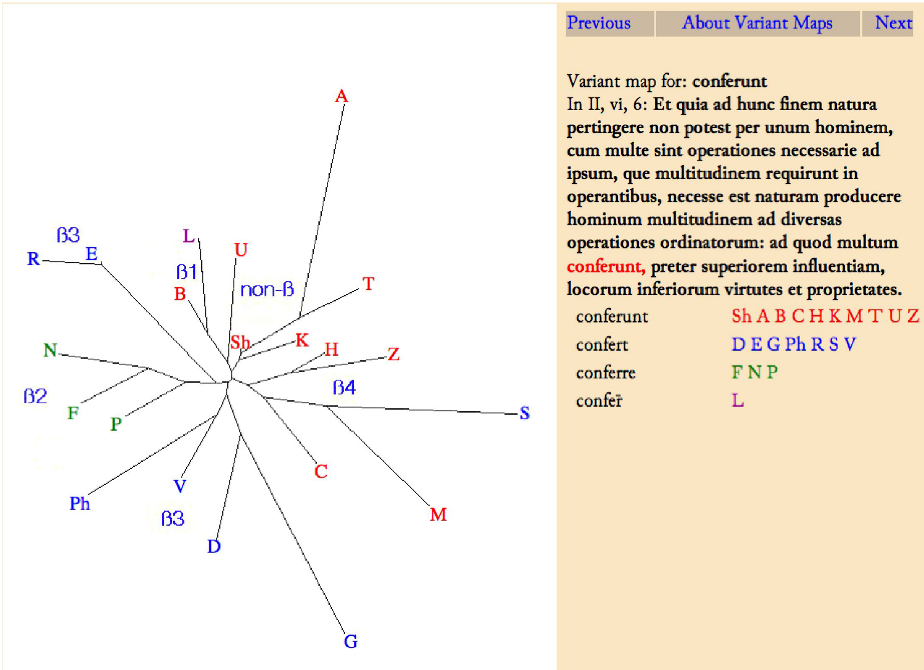


Fig. 4. The unrooted phylogram for the witnesses to Dante's *Monarchia*, generated directly from the computer-assisted collation

variants, as a simple matter of saying 'this phrase in this manuscript corresponds with that phrase in that manuscript'. Look, for example, at this set of variants:

He was agast so
He was agast
So he was agast
He was so agast
He was agast and feerd
So was he agast

Now, Collate will do an excellent job of letting you see each of these phrases as variants of one another: in essence, of each phrase being a variant of each other, thus:

He was agast so] He was agast; So he was agast; He was so agast; He was agast and feerd; So was he agast

You can see, immediately, that there is something wrong here. The Collate presentation makes it look as if each variant is equally different from every other one. But clearly, this is not the case. You can see that the first, third and fourth differ only in the postion of the the word 'so'. Then, you can see that the second differs from these three by dropping the word 'so', while the fifth differs from the second

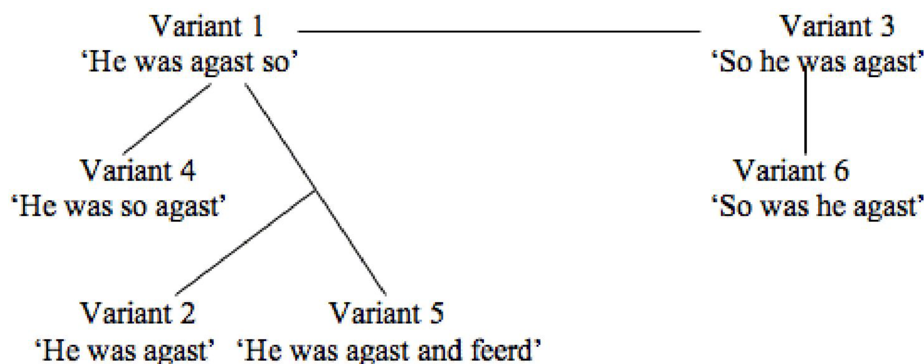


Fig. 5. Schema of the variants on this phrase, showing the relative closeness of each variant

by adding the phrase 'and feerd'. Finally, the sixth one has the word so in the same position as the third one, but changes the order 'he was' to 'was he'.

In fact, what we would like to see is something like Figure 5.

But Collate, as it is, just won't let you do this. A significant by-product of this is that Collate handles transpositions rather incompletely. Take this variation:

he was] was he

Collate identifies the phrase variation nicely. But Collate does not tell us what could be rather crucial information: that both witnesses do have the same two words, just in a different order.

What I have just described is known in genomic comparison systems as 'multiple progressive alignment': that is, you build up the picture of comparison piece by piece. [2] I think it is very important that a successor to Collate contains this facility.

3 Towards Collaborative Scholarly Editing

So far, the two good things I have described that Collate does and the one good thing it does not. I said too that we now realize that in one respect, the design of Collate is fundamentally flawed. It works like this. When I wrote Collate, I was a typical lone scholar, making an edition on my own. I found all the manuscripts, I transcribed them, I collated them, I edited the text. I did it all myself and I liked it that way. Further, every other editor I knew then worked the same way. So it is no surprise that the first and all subsequent versions of Collate were built with the same model in mind: that is, I ran the program on my computer; all the files were on my computer; everything was under my command.

At the time, I thought this was a virtue. But now, I can see that this is the biggest single defect of Collate. Quite simply, it was never designed as a

[2] See the article by Matthew Spencer and Chris Howe on the use of multiple progressive alignment in evolutionary biology and its possible application to scholarly collation [13].

collaborative tool and it is very awkward to use the program in a collaborative situation. This did not matter, at all, when scholars did not collaborate. But the single greatest effect of the digital revolution, in my opinion, is not that it is giving us wonderful digital libraries, with instant access to everything we want to see, or that it is giving us wonderful new publication possibilities, or that it offers all kinds of marvellous tools - databases, analytic programs, and more. In my opinion, the single greatest effect of the digital revolution is that it it is empowering a new model of collaboration, and hence new modes of readership and study, among scholars, and between scholars and readers. Through well-constructed scholarly networks over the web, scholars and readers may not only look at materials: they may make them, annotate them, correct them, draw conclusions from them and then contribute to others their conclusions. Further, this may happen near-simultaneously: a library may contribute manuscript images in the morning; by midday a scholar has identified the text; by mid-afternoon a knowledgeable reader has transcribed it; in the evening, another scholar has collated this new transcript against other versions of the text, in other manuscripts. Nor is there any need for the scholars and readers to have any formal affiliation: they may be working far apart, without any project or other framework beyond common access to the web, shared interest and expertise.

The relevance of this to any large scale editorial project – indeed, to any editorial project at all – is obvious. Imagine that all the Sanskrit scholars of the world work in a single online workspace. In this workspace, some of you transcribe manuscripts; others of you collate the transcripts; others analyse the results of the collation. For some time I have been describing, in various articles and talks, what I have described as 'distributed, dynamic and collaborative editions'. The concept is not that there is a single system, a single set of software tools, which everybody uses. Instead, across the web we have a federation of separate but co-operating resources, all within different systems, but all interlinked so that to any user anywhere it appears as if they were all on the one server. To take the *Canterbury Tales* as an example: there might be transcriptions of the different manuscripts of the first line of the Tales available on different servers, made by different scholars, in New York, in Birmingham, in Utah. I can, any scholar can, access all these simultaneously: alongside images of the manuscripts, with collations, analyses, much else. Remarkably, we need very little to make this work: in fact, all we need are some basic agreements on how we name things, and on how we pass information between ourselves. We have much of this already, in the form of various metadata, encoding and web services protocols. We certainly have all the hardware and software we need, and more than enough people with all the skills needed to make this happen. I confess I do feel some frustration with the various funding agencies who could set us well along the way by giving just a little bit of help. Instead, they appear to be fixated with Grand Single Solutions: I am thinking of the sort of projects funded by the NSF in America, and eScience in the UK, and by the Mellon Foundation, which show a predeliction towards grandiose projects emenating from prestigious institutions, Bamboo and SEASR being egregious instances of the genre [14][15]. Let me say this clearly, as most

scholars seem afraid to say it: projects like these are vast wastes of time, effort and money. But actually, I don't think we need this funding – I think the idea is so good, and so obvious, that it will take off with very little or no funding.

I have, in other places, commented on what seems to me a very strange development [17][18][19]. Over the last fifteen years it has actually become harder for an ordinary scholar to create a high-quality scholarly edition in digital form. Indeed, it has become so much harder that a number of scholars and editorial projects have turned away from the digital medium: a development which really ought to alarm us. The answer to this flight from the digital is rather simple: we should make it as easy, or even easier, for a scholar to make a high-quality digital edition as it is to make a print edition. It seems to me quite absurd that, several decades into the digital revolution and with all the funding and all the effort lavished on it, we have not reached that point. We know that there are many scholars and readers with the interest and expertise to contribute greatly to the shared endeavour that is textual scholarship, in the editing of Sanskrit, of biblical materials, of medieval vernacular classics, everywhere there is text. The digital world should provide a space where any scholar with something useful to contribute may do so; where all may gain from the wealth of information so created.

You may ask: what am I doing to bring about this millennial vision? Several things. Firstly, I am a partner in an EU project named INTEREDITION, founded to bring about the creation of a supra-national infrastructure for digital textual scholarship [16]. The first workshop of INTEREDITION in September 2008 will address these issues. Even more immediately, the Birmingham and Münster partners in various New Testament editing projects are setting up what we call a Virtual Manuscript Room: a shared workspace in which all the many editors in these projects may work, as I have outlined it here. In our implementation of the Virtual Manuscript Room, one of the key factors will be a new version of our Anastasia software, redesigned so that she works directly from an XML database as the backbone of our live, interactive shared workspace. The first stage of this project has now received funding from the UK Joint Information Systems Committee [20]. Most immediately of all: since January a group of Dutch and German scholars have been working with me on what we call CollateX: the much-wished for successor to Collate.

I am suggesting here the future lies with a network of many servers, all holding different parts of an edition, with many other servers providing a range of services to the readers and scholars interested in this edition. Here, as a taste of what may come, is a tool we are making. We now have many manuscript images on the web. We now have many transcriptions of these same manuscripts on the web. The obvious thing to do is to align the text in the image with the text in the transcript. Here is a case where we have actually done exactly this, in the online version of Codex Sinaiticus [21], shown in Figure 6.

You can see that clicking on the word in the transcription highlights this word in the image.

We can imagine a vast number of potential research possibilities coming from this tool. For example: one could apply OCR software and pattern matching

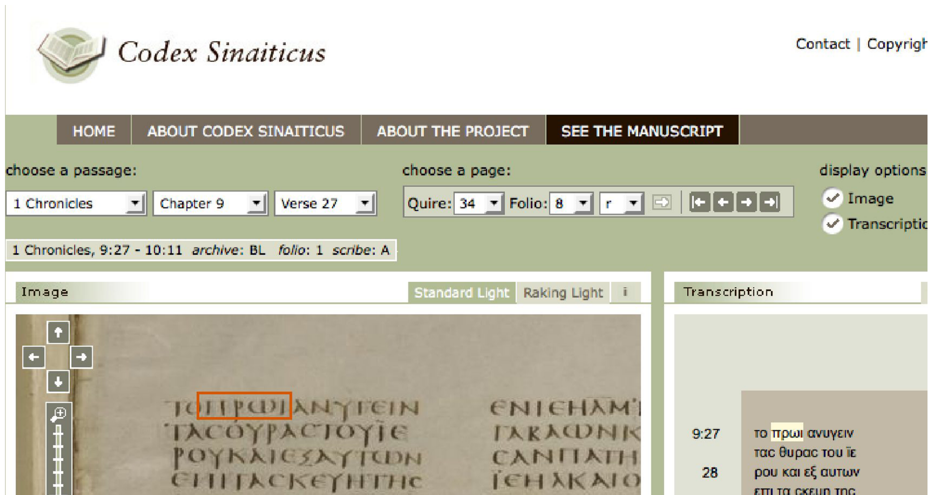


Fig. 6. Codex Sinaiticus on the web, showing the alignment of text and transcription

techniques to correlate every letter in the image with those in the manuscript, to explore questions of the characteristics of different scribal hands. One could also use automated analysis of the ruling and layout of the manuscript leaves to explicate its construction.

Perhaps best of all: tools like this will enable other people to do research we can not even imagine. For myself, I can not think of anything better.

References

1. The Collation and Textual Criticism of Icelandic Manuscripts (1): Collation (2): Textual Criticism. Literary and Linguistic Computing 4, 99–104, 174–181 (1989)
2. Collate. Computer Program. Release 1.0, April 1991; revised versions 1992-2008. Oxford: Computers and Manuscripts Project; Leicester: Centre for Technology and the Arts; Birmingham Institute for Textual Scholarship and Electronic Editing
3. Shaw, P.: Dante's Monarchia. Scholarly Digital Editions, Birmingham 2005 (accessed September 9, 2008), `http://www.sd-editions.com/Monarchia`
4. The Institute for New Testament Textual Research, Münster (accessed September 9, 2008), `http://www.uni-muenster.de/NTTextforschung/`
5. The Institute for Textual Scholarship and Electronic Editing, Birmingham (accessed September 9, 2008), `http://itsee.bham.ac.uk`
6. Wasserman, T.: The Epistle of Jude: Its text and transmission. Coniectanea Biblica New Testament Series - CBNTS 43, Almqvist and Wiksell (2006)
7. The Canterbury Tales Project (accessed September 9, 2008), `http://www.canterburytalesproject.org`
8. The Cancioneros Project (accessed September 9, 2008), `http://cancionerovirtual.liv.ac.uk/`
9. The William of Ockham Project (accessed September 9, 2008), `http://www.britac.ac.uk/pubs/dialogus/ockdial.html`

10. The JUXTA project (accessed September 9, 2008),
http://www.juxtasoftware.org/
11. The EDMAC macors (accessed September 9, 2008),
http://www.ucl.ac.uk/~ucgadkw/edmac/
12. Robinson, P.M.W. (ed.): The Miller's Tale on CD-ROM. Scholarly Digital editions, Leicester (2004)
13. Spencer, M., Howe, C.: Collating Texts Using Progressive Multiple Alignment. Computers and the Humanities, 253–270 (2004)
14. The BAMBOO project (accessed September 9, 2008),
http://www.projectbamboo.org/
15. The SEASR project (accessed September 9, 2008), http://seasr.org/
16. The INTEREDITION project (accessed September 9, 2008),
http://interedition.huygensinstituut.nl/
17. Robinson, P.M.W.: Current issues in making digital editions of medieval texts–or, do electronic scholarly editions have a future? Digital Medievalist 1.1 (2005),
http://www.digitalmedievalist.org/article.cfm?RecID=6
18. Robinson, P.M.W.: Where We Are with Electronic Scholarly Editions, and Where We Want to Be. Jahrbuch für Computerphilologie (January 2004) (In print in Jahrbuch für Computerphilologie 2004, 123–143),
http://computerphilologie.uni-muenchen.de/ejournal.html
19. Robinson, P.M.W.: Current Directions in the Making of Digital Editions: towards interactive editions. Ecdotica 4, 176–191 (2007)
20. The Virtual Manuscript Room (accessed September 9, 2008),
http://www.itsee.bham.ac.uk/projects/vmr/index.htm
21. Codex Sinaiticus on the web (accessed September 9, 2008),
http://www.codex-sinaiticus.net/en/manuscript.aspx

Critical Edition of Sanskrit Texts*

Marc Csernel[1] and François Patte[2]

[1] INRIA, Projet AXIS,
Domaine de Voluceau, Rocquencourt BP 105
78153 Le Chesnay Cedex, France
Marc.Csernel@inria.fr
[2] UFR de Mathématiques et Informatique,
Université Paris Descartes,
45 rue des Saints-Pères,75270 Paris cedex 06, France
Francois.Patte@math-info.univ-paris5.fr

Abstract. A critical edition takes into account all the different known versions of the same text in order to show the differences between any two distinct versions. The construction of a critical edition is a long and, sometimes, tedious work. Some software that help the philologist in such a task have been available for a long time for the European languages. However, such software does not exist yet for the Sanskrit language because of its complex graphical characteristics that imply computationally expensive solutions to problems occurring in text comparisons.

This paper describes the Sanskrit characteristics that make text comparisons different from other languages, presents computationally feasible solutions for the elaboration of the computer assisted critical edition of Sanskrit texts, and provides, as a byproduct, a distance between two versions of the edited text. Such a distance can then be used to produce different kinds of classifications between the texts.

1 Introduction

A critical edition is an edition that takes into account all the different known versions of the same text. If the text is mainly known through a great number of manuscripts that include non trivial differences, the critical edition often looks rather daunting for readers unfamiliar with the subject:

– If the number of texts to compare is small and differences between texts are not too great, the text looks just like any commentated editions.
– If the text is mainly known through a great number of manuscripts that include non trivial differences, the critical edition looks often rather daunting for readers unfamiliar with the subject. The edition is then formed mainly by footnotes that enlighten the differences between manuscripts, while the main text (that of the edition) is rather short, sometimes a few lines on a page.

* This paper was supported by the ACI CNRS "histoire des savoirs" and the Asia IT & C contract 2004/091-775.

G. Huet, A. Kulkarni, and P. Scharf (Eds.): Sanskrit CL 2007/2008, LNAI 5402, pp. 358–379, 2009.

Note that in either case, the main text is established by the editor through his own knowledge. More explicitly, the main text can be either a particular manuscript, or a "main" text, built according to some specific criteria chosen by the editor.

Building a critical edition by comparing texts two by two, especially manuscript ones, is a task which is certainly long and, sometimes, tedious. This is why, for a long time, computer programs have been helping philologists in their work (see [1] or [2] for example), but most of them are dedicated to texts written in Latin (sometimes Greek) scripts. For example, the Institute for New Testament Textual Research , at Münster University, provides an interactive critical edition of the Gospels [3].[1].

In this paper we focus on the critical edition of manuscripts written in Sanskrit.

Our approach will be based on and illustrated by paragraphs and sentences that are extracted from a collection of manuscripts of the "Banaras gloss", *kāśikāvṛtti* in Sanskrit (Kāśi is the name of Banaras). The Banaras gloss was written around the 7th century A.D., and is the most widespread, the most famous, and one of the most pedagogical commentary on the notorious Pāṇini grammar.

Pāṇini's grammar is known as the first **generative** grammar and was written around the fifth century B.C. as a set of rules. These rules cannot be understood without the explanation provided by a commentary such as the *kāśikāvṛtti*. Notice that, since some manuscripts have been damaged by mildew, insects, rodents…, they are not all complete. In particular, they do not include all chapters; generally around fifty different texts are available for comparison at the same time.

In what follows we will first describe the characteristics of Sanskrit that matter for text comparison algorithms as well as for their classification. We will also present briefly the textual features we use to identify and to quantify the differences between manuscripts of the same Sanskrit text. We will show that such a comparison requires to use a lemmatized text as the main text.

Roughly speaking, lemmatization is a morpho-linguistic process which makes each word appear in its base form, generally followed by a suffix indicating its inflected form. For example *walking*, consists of the base form *walk*, followed by the suffix *ing* which indicates the continuous form. After a lemmatization each word will, at least, appear as separated from the others.

The revealed differences, which as a whole, form one of the most important parts of the critical edition, provide all the information required to build distances between the manuscripts. Consequently we will build phylogenetic trees assessing filiations between them, or any kind of classification regrouping the manuscripts into meaningful clusters. Finally, we will discuss the definition of a method of computation of faithful distances between any two Sanskrit texts, provided one of them is lemmatized.

2 How to Compare Sanskrit Manuscripts

2.1 Sanskrit and Its Graphical Characteristics

One of the main characteristic of Sanskrit is that it is not linked to a specific script. A long time ago Sanskrit was mostly written with the Brāhmī script, but nowadays

[1] Critical editions of the Gospels have induced a considerable amount of studies.

Devanāgarī is the most common one. Other scripts may be used, such as Bengali, in northern India, or Telugu, in southern India. In Europe, an equivalent (but fictive) situation would be to use either the Latin, Cyrillic, or Greek alphabets to write Latin. Sanskrit is written mostly with the Devanāgarī script that has a 48 letter alphabet.

Due to the long English presence in India, a tradition of writing Sanskrit with the Latin alphabet (a transliteration) has been established for a long time by many European scholars such as Franz Bopp in 1816. The modern IAST — International Alphabet of Sanskrit Transliteration — follows the work of Monier-Williams in his 1899 dictionary[4]. All these transliteration schemes were originally carried out to be used with traditional printing. It was adapted for computers by Frans Velthuis [5], more specifically to be used with TEX. According to the Velthuis transliteration scheme, each Sanskrit letter is written using one, two or three Latin characters; notice that according to most transliteration schemes, upper case and lower case Roman characters have a very different meaning. In this paper, unless otherwise specified, a letter is a Sankrit letter represented, according to the Velthuis scheme, by one, two or three Latin characters.

In ancient manuscripts, Sanskrit is written without spaces, and from our point of view, this is an important graphical specificity, because it increases greatly the complexity of text comparison algorithms. One may remark that Sanskrit is not the only language where spaces are missing in the text: Roman epigraphy and European Middle Age manuscripts are also good examples of that.

2.2 The Different Comparison Methods

Comparing manuscripts, whatever the language, can be achieved in two ways:

- When building a critical edition, the notion of word is central, and an absolute precision is required. For example, the critical edition must indicate that the word *gurave* is replaced by the word *gaṇeśāya* in some manuscripts, and that the word *śrī* is omitted in others.
- When establishing some filiation relations between the manuscripts, or for a classification purpose, the notion of word can be either ignored, or taken into account. The only required information is the one needed to build a distance between texts. Texts can be considered either as letter sequences, or as word sequences.

Considering each text as a letter sequence, Le Pouliquen [6] proposed an approach that determines the so called *"Stemma codicum"* (nowadays *filiation trees*) of a set of Sanskrit manuscripts. The first step consists in the construction of a distance according to the Gale and Church algorithm [7]. This algorithm was first developed to provide sentence alignments in a multi-lingual corpus, for example a text in German and its English translation. It uses a statistical method based on sentence length. Gale and Church showed that the correlation between two sentence lengths follows a normal distribution. Once the distance is computed, a phylogenetic tree is built using the N-J —Neighbour-Joining— algorithm ([8]).

On the other hand, each critical edition deals with the notion of word. Since electronic Sanskrit lexicons such as the one built by Huet [9,10] do not cope with grammatical texts, we must find a way to identify each Sanskrit word within a character string, without the help of either a lexicon or of spaces to separate the words.

2.3 How Shall We Proceed?

The solution comes from the lemmatization of one of the two texts: the text of the edition. The lemmatized text is prepared **by hand** by the editor. We call it a *padapāṭha*, according to a mode of recitation where syllables are separated.

From this lemmatized text, we will build the text of the edition, that we call a *saṃhitapāṭha*, according to a mode of recitation where the text is said continuously. The transformation of the *padapāṭha* into the *saṃhitapāṭha* is not straightforward because of the existence of *sandhi* rules.

What is called *sandhi* — from the Sanskrit: liaison — is a set of phonetic rules which apply to the morpheme junctions inside a word or to the junction of words in a sentence. Though these rules are perfectly codified in Pāṇini's grammar, they could become quite tricky from a computer point of view. For instance, the final syllable *as* is mostly changed into *o* if the next word begins with a voiced letter, but the word *tapas* (penance) becomes *tapo* when it is followed by the word *dhana* (wealth) to build the compound *tapodhana* (one who is rich by his penances), while it remains *tapas* when composed with the suffix *vin*: *tapasvin* (an ascetic). What is a rule for Pāṇini, becomes an exception for computer programs and we have to take this fact into account.

A text with separators (such as spaces) between words, can look rather different (the letter string can change greatly) from a text where no separator are found.

We call the typed the text, corresponding to each manuscript, a *mātṛkāpāṭha*. Each *mātṛkāpāṭha* contains the text of a manuscript and some annotation commands.

Table 1. The collation commands

`\gap`	Gap left intentionally by a scribe	`\deleted`	Text deleted by the scribe
`\afterc`	The text after a scribe's correction.	`\beforec`	The text before a scribe's correction
`\scribeadd`	Insertion made by the scribe without the presence of gap	`\eyeskip`	The scribe copying the text has skipped his eyes from one word to the same word later in the text.
`\doubt`	Text is not easily readable	`\inferred`	Text very difficult to read
`\lacuna`	The text is damaged and not readable	`\illegible`	Mainly concerns the deleted text
`\insertioningap`	Insertion made by a scribe in a gap	`\foliochange`	
`\ignoredtext`	This text of the manuscript, is not part of the opus	`\marginote`	Insertion made by the scribe, as his own commentary (but not part of the text)
`\notes`	Notes made by the scholar in charge of the collation		

These commands allow some information from the manuscript to be taken into account, but this information is not part of the text, such as ink colour, destruction, etc. They provide a kind of meta-information.

The typing of each *mātṛkāpāṭha* is done by scholars working in pair, one reading, one typing (alternatively). To avoid the typing of a complete text, they copy and modify the text of the *saṃhitapāṭha* according to the manuscript.

- **First step:** A twofold lexical preprocessing. First the *padapāṭha* is transformed into a virtual *saṃhitapāṭha* in order to make a comparison with a *mātṛkāpāṭha* feasible.

  The transformation consists in removing all the separations between the words and then in applying the *sandhi*. This virtual *saṃhitapāṭha* will form the text of the edition, and will be compared to the *mātṛkāpāṭha*. As a sub product of this lexical treatment, the places where the separation between words occurs will be kept into a table which will be used in further treatments (see Sec.: 4.4).

  On the other hand, the *mātṛkāpāṭha* is also processed, the treatment consists mainly in keeping the collation commands out of the texts to be compared. The list of the commands can be found in Table 1 (p. 361) with some explanation when needed. Notice that for practical reasons, these commands cannot, for the time being, be nested. Out of all these commands just a few have an incidence on the texts to be compared.
- **Second step:** An alignment of a *mātṛkāpāṭha* and the virtual *saṃhitapāṭha* (an alignment is an explicit one to one correspondence of the letters of the two texts.) A more precise definition can be found on page 368. The Longest Common Subsequence algorithm is applied to these two texts. The aim is to identify, as precisely as possible, the words in the *mātṛkāpāṭha*, using the *padapāṭha* as a pattern. Once the words of the *mātṛkāpāṭha* have been determined, we can see those which have been added, modified or suppressed.

The comparison is done paragraph by paragraph, the different paragraphs being constructed in each *mātṛkāpāṭha* by the scholar who collated them, according to the paragraph made in the *padapāṭha* during its elaboration. In a first stage, the comparison is performed on the basis of a Longest Common Subsequence. Each of the obtained alignments, together with the lemmatized text (i.e. the *padapāṭha*), suggests an identification of the words of the *mātṛkāpāṭha*. However, due to the specificities of Sanskrit, the answer is not straightforward, and a consistent amount of the original part of this work concerns this identification process. Surprisingly the different rules used for this determination are not based on any Sanskrit knowledge, but on common sense. The result of the application of these rules has been validated by Sanskrit philologists.

We remark that the kind of results expected for the construction of a critical edition (what words have been added, suppressed or replaced in the manuscript) is similar to the formulation of an edit distance, but in terms of *words*. The results we obtain from the construction of the critical edition can be transformed into a distance between the manuscripts.

3 The Lexical Preprocessing

The goal of this step is to transform both the *padapāṭha* and the *mātṛkāpāṭha* in order to make them comparable. This treatment will mainly consist in transforming the *padapāṭha* into a *saṃhitapāṭha*. The *mātṛikāpāṭha* will be purged of all collation commands, except some of the commands which modify the text to be compared, namely `\scribeadd`, `\afterc`, `\inferred`. All lexical treatments are build using `Flex` [11], a `Linux` version of `Lex` [12] which is a free and widely known software.

At the end of the lexical treatment the text corresponding respectively to the *padapāṭha* and the *mātṛkāpāṭha* is transmitted to the comparison module with an internal encoding (see Table 4, p. 366). This allows us to ensure the comparison whatever the text encoding — unicode instead of Velthuis code for instance — the only condition is to build a new lexical scheme, which is a perfectly delimited work albeit a bit time-consuming.
An example of *padapāṭha*:

```
iti+anena krame.na var.naan+upa^di"sya+ante .na_kaaram+itam+|
```

we can see that words are separated by spaces and three different lemmatization signs: +, _, ^ which have the following meanings:

– **+**: Indicates a separation between inflected items in a sentence.
– **_**: Indicates a separation between non inflected items of a compound word.
– ^: Indicates the presence of a prefix; this sign is not, for the moment, taken into account for the comparison process. It will be used for a future automatic index construction.

3.1 The Lexical Preprocessing of the *mātṛkāpāṭha*

The main goal of this step is to remove the collation commands in order to keep only the text of the manuscript for a comparison with the *saṃhitapāṭha*. The list of these commands can be found in Table 1 (p. 361). The tables described hereafter follow more or less the `Lex` syntax, with a major exception, for readability reason: the suppression of the protection character denoted " \". We will note briefly some of the main features:

The character "|" means **or**; a name included within braces, such as `{VOWEL}`, is the name of a letter subset defined in Table 2 (p. 364). It can be replaced by any letters of the subset. The character "/" means **followed by**, but the following element will not be considered as part of the expression: it will stay within the elements to be further examined; examples of the use of the character "/" will be found hereafter in Table 3 (p. 364).

Note that some possible typographical errors induced us to remove all the spaces from the *mātṛkāpāṭha* before the comparison process. Thus no words of the *mātṛkāpāṭha* can appear separately during that process.

Table 2 provides a definition for the subset definition such as `VOWEL_A` defined in the third line, which is a subset of the alphabet containing all the vowels except **a**, in fact one of the following letters:

aa, i, ii, u, uu, .r, .R, .l, .L, e, ai, o, au

Table 2. Some lexical definition of letter categories

<table>
<tr><td>SOUR</td><td>k|kh|c|ch|.t|.th|t|th|p|ph|"s|.s|s|.h</td></tr>
<tr><td>NAS</td><td>n|.n|"n|~n|m|.m</td></tr>
<tr><td>VOWEL_A</td><td>aa|i|ii|u|uu|.r|.R|.l|.L|e|ai|o|au</td></tr>
<tr><td>VOWEL</td><td>a|{VOWEL_A}</td></tr>
<tr><td>DIPH</td><td>e|ai|o|au</td></tr>
<tr><td>CONS</td><td>k|kh|g|gh|"n|c|ch|j|jh|~n|.t|.th|.d|.dh|
.n|t|th|d|dh|n|p|ph|b|bh|m|"s|.s|s</td></tr>
<tr><td>SON</td><td>g|gh|j|jh|.d|.dh|d|dh|b|bh|l|r|y|v|{NAS}|.m|h</td></tr>
<tr><td>GUTT</td><td>k|kh|g|gh|"n</td></tr>
<tr><td>PALA</td><td>c|ch|j|jh|~n</td></tr>
<tr><td>LEMM</td><td>+|_|^</td></tr>
<tr><td>DENTA</td><td>t|th|d|dhn</td></tr>
<tr><td>LABIA</td><td>p|ph|b|bh|m</td></tr>
</table>

according to the Velthuis encoding scheme, and `VOWEL`, next line, defined by any letter in: `a|{VOWEL_A}` can be any letter in `VOWEL_A` or `a`. Notice that the subset `LEMM` contains the different lemmatization signs found in the *padapāṭha*.
Table 3 (p. 364) contains some example of **generative** *sandhi* where a new letter (or a sequence of letters) is inserted within the text. Table 5 (p. 366) contains some examples of ordinary *sandhi* where a set of letters is replaced by one or two other letters.
The contents of both preceding tables will be explained in the following section.

3.2 The Lexical Preprocessing of the *padapāṭha*

The main goal of this step is to apply the *sandhi* rules in order to transform the *padapāṭha* into a *saṃhitapāṭha*, the other goal is to purge the *padapāṭha* of all unwanted characters. The *sandhi* (p. 361) are perfectly determined by the grammar of Sanskrit (see for example [13]). They induce a special kind of difficulties due to the fact that their construction can be, in certain cases, a two-step process. During the first step, a *sandhi* induces the introduction of a new letter (or a letter sequence). This new letter can induce, in the second step, the construction of another *sandhi*. The details of the lexical transformation expressed as a `Flex` expression can be found in Table 3 (p. 364) for the first step, and in Table 5 (p. 366) for the second one.

Table 3. Some of the generative *sandhi*

<table>
<tr><td>as+/{SON}</td><td>Add("o"); AddSpace();</td></tr>
<tr><td>as+/{VOWEL_A}</td><td>Add("a"); AddSpace();</td></tr>
<tr><td>as+a</td><td>Add("o.a");</td></tr>
<tr><td>aas+/{VOWEL}</td><td>Add("aa"); AddSpace();</td></tr>
<tr><td>as+/(k|p|s|.s|"s)</td><td>Add("a.h"); AddSpace();</td></tr>
<tr><td>ai/+{VOWEL}</td><td>Add("aa"); AddSpace();</td></tr>
<tr><td>ai(_|^)/{VOWEL}</td><td>Add("aay");</td></tr>
</table>

Table 3 can be read in the following way: the left part of the table contains a Flex expression, the right part some procedure calls. The two procedures are Add("xxx"), which adds the letter sequence xxx to the text of the *padapāṭha*, and AddSpace() which adds a space within the text. When the expression described in the left part is found within the *padapāṭha*, the procedures described in the right part are executed. The letters belonging to the expression on the left of the sign "/" are removed from the text. The different expressions of the left part are checked according to their appearence. The tests are done in sequential arrangement.

For example the first three lines of Table 3 state that:

- If a sequence as, followed by a lemmatization sign +, is followed by any letter of the {SON} subset defined in Table 2, the program puts in the text an o followed by a space; the letter sequence as+ will be dropped out from the text, but not the element of {SON}.
  Example: If the sequence bahavas+raa"sayas+hataas+| is found in the *padapāṭha*, the sequence as+/{SON}: bahavas+r and raa"sayas+h[2] is found twice.
  Therefore, according to the rules defined in the right column of the table, we get as a result in the *saṃhitapāṭha*: bahavo␣raa"sayo␣hataaḥ|, corresponding to the Sanskrit text: *bahavo rāśayo hatāḥ* |
- If the sequence as+ is followed by a letter which belongs to {VOWEL_A}, an a will be generated and the element belonging to {VOWEL_A} will remain.
  Example: If the sequence prakalpitas+i.s.taraa"sis+| is found in the *padapāṭha*, we have one sequence as+/{VOWEL_A}: prakalpitas+i and, in this case, the program will return within the *saṃhitapāṭha*: prakalpita␣i.s.taraa"si.h|. The Sanskrit text: *prakalpita iṣṭarāśiḥ* |
- If the sequence as is followed by a lemmatization sign + and by the letter a, it will be replaced by the sequence o.a and no space will be added.
  Example: If the sequence yogas+antare.nonayutas+ardhitas+| is found in the *padapāṭha*, the sequence appears twice: yogas+a and yutas+a; this will be changed into:
  yogo.antare.nonayuto.ardhita.h|, corresponding to the Sanskrit text: *yogo'ntareṇonayuto'rdhitaḥ*|

Once Table 3 has been used with the *padapāṭha*, Table 5 and Table 4 are used in the same lexical pass.

Table 4 is really simple: the left part contains a character sequence corresponding to the Velthuis code, the right part contains a return code followed by an upper case letter sequence beginning by an L. This letter sequence is the name of an internal code that corresponds to a *devanāgarī* letter and will be used for further treatment.

Table 5 is a little bit more complicated in its right part. It contains references to two variables Alter and Next and each of these variables is affected by a value of the internal code corresponding to the Velthuis code.

[2] The case hataas+| is not one of these for two reasons: 1) aas+ is different from as+, according to the Velthuis encoding scheme, 2) | is a punctuation mark and does not belong to the category {SON}, it has its own way of treatment.

Table 4. Examples of Velthuis characters encoding, with linked internal code

e	return LE;
ai	return LAI;
aa	return LABAR;
k	return LK;
"n	return LNQU;
~n	return LNTI;
.n	return LNPO;

Table 5. Some normal *sandhi*

.m/{GUTT}	Alter = LNQU; return LMPO;
.m/{PALA}	Alter = LNTI; return LMPO;
.m/{DENTA}	Alter = LN; return LMPO;
.m/{LABIA}	Alter = LM; return LMPO;
(a|aa){LEMM}(a|aa)	return LABAR;
(a|aa){LEMM}(o|au)	return LAU;
.r/{LEMM}({VOWEL}|{DIPH})	return LR;
e{LEMM}a	Next = LAVA; return LE;
o{LEMM}a	Next = LAVA; return LO;
(k|g)/{LEMM}{SOUR}	return LK;
(k|g|c)/{LEMM}({SON1}|{VOWEL})	return LG;
(k|g|c)/{LEMM}{NAS}	Alter = LNPO; return LG;
(.t|.d|.s)/{LEMM}{SOUR}	return LTPO;
(.t|.d|.s)/{LEMM}{NAS}	Alter = LNPO; return LDPO;
(.t|.d|.s)/{LEMM}({SON}|{VOWEL})	return LDPO;
as/+" "	Next = LHPO; return LA;

The variable `Alter` corresponds to an alternate value to the returned code (in other terms, the code of another possible letter), the variable `Next` corresponds to the code letter generated by the *sandhi* which will **always** follow the returned letter. If `Alter` take a value, the letter is equivalent to the letter returned by the normal process so, the returned and the `Alter` value can be exchanged and the distance between the letters is zero.

The first four lines treat the letter `.m` — *ṃ, anusvāra* — in different contexts: if this letter is followed by a letter belonging to one of the subsets `GUTT`, `PALA`, `DENTA`, `LABIA`, defined in Table 2, there could be, in some manuscript, an alternate letter for it. This is mainly due to scribe habits and we must make the software aware of this. For instance the word *aṅka* can also be written *aṃka*, in which case we are in the situation `.m/GUTT`; according to the instruction: `Alter=LNQU; return LMPO;`, while comparing our virtual *saṃhitapāṭha* with a *mātṛkāpāṭha*, if, at the same place in two *mātṛkāpāṭha*, the comparison software reads `a.mka` or `a"nka`, no variant will be reported and the value of the distance distance between `a.mka` or `a"nka` is zero. A similar situation occurs for the readings `pa.n.dita/pa.m.dita` (`.m/PALA`) or `sandhi/sa.mdhi` (`.m/DENTA`) or `sambhuu/sa.mbhuu` (`.m/LABIA`).

The variable `Next` is used whenever the *sandhi* rule induces the production of a new character next to the character (or string) concerned by the *sandhi*.

For instance, if we have: `tanmuule+a.s.tayute`, the `e` before the lemmatization sign will remain, but the `a` will be elided and replaced by an *avagraha*; this is the meaning of the rule in line 8: if we have `e{LEMM}a` then `e` is kept: `return LE` and next to it an *avagraha* is produced: `Next=LAVA`; so we get: `tanmuule.a.s.tayute`.

The same procedure is done with the last line of the table: if a word is ended by `as` and followed by a blank space, `as` is dropped (meaning of "`/`"), `a` is returned followed by a *visarga*: `Next=LHPO`.

Line 5 contains the premises of further difficulties: it states that the letter `a` or `aa` followed by a lemmatization sign and the by the letter `a` or the letter `aa` (correponding to the sanskrit letter *a* and *ā* in traditional transliteration) will become the `LABAR` code (corresponding to the letter `aa`: *ā*). Two letters and the lemmatization sign will become a single letter. Consequently, if a variant occurs which concerns the letter `aa` the program will not know if the variant concerns the word of the *padapāṭha* before or after the lemmatization sign.

First example. If we have in the *padapāṭha*: `"sabda_artha.h`, it will become `"sabdaartha.h` in the *saṃhitapāṭha*. If we have in the *mātṛkāpāṭha*: `sabde.artha.h`, the program will have to decide between some of the following possible solutions:

`"sabda` has been changed into `"sabde` and `artha.h` has been changed in `.artha.h`
`"sabda` has been changed into `"sabd` and `artha.h` has been changed in `e.artha.h`
`"sabda` has been changed into `"sabde.a` and `artha.h` has been changed in `rtha.h`

Second example. If we have the *padapāṭha*: `asya+artha.h`, it will become `asyaartha.h` in the *saṃhitapāṭha*. If we have in the *mātṛkāpāṭha*: `asyaa artha.h`, as the program, in the lexical preprocessing, removes the spaces in the *mātṛkāpāṭha*, it will have to decide between some of the following possible solutions:

`asya` has been changed into `asyaa` and `artha.h` stays unchanged.
`asya` has been changed into `asyaa` and `artha.h` has been changed in `rtha.h`.

Third example. If we have the *padapāṭha*:
`iti+u_kaare.na+a_kaara_aadaya.h`, it will come in the *saṃhitapāṭha* as `ityukaare.naakaaraadaya.h`.
If we have in the *mātṛkāpāṭha*: `ityukaare.nekaaraadaya.h`, the comparison can be very confusing because one word is completely missing: the `a` in `a_kaara`. This creates a very important problem: we had not imagined at the beginning of our work that a complete word could disappear if only one letter was missing.

4 Comparing the Sanskrit Manuscripts with the Text of the Edition

In this section we will come to the heart of our research. We compare, sentence by sentence, the text of each *mātṛkāpāṭha* (i.e. a collated manuscript), purged of every collation commands, with the *padapāṭha* transformed into a *saṃhitapāṭha*.

Table 6. Examples of possible alignments

a	a	a	_	b	b	b
a	a	a	c	_	b	b

a	a	a	_	b	b	b
a	a	a	c	b	_	b

a	a	a	b	b	_	b
a	a	a	_	c	b	b

For each comparison we start to **align** each *mātṛkāpāṭha* sentence, word by word, with those of the *saṃhitapāṭha*. This comparison uses the word limits provided by the lemmatization done in the *padapāṭha*. We use a basic tool: the Longest Common Subsequence (LCS) algorithm to begin our alignment process.

In the following, we describe the LCS algorithm by giving an example. Then we explain why the use of the LCS still raises some problems. We can solve some on these problems by carefully sailing through the LCS matrix, thanks to the limits provided by the *padapāṭha*.

Even with such a help, and a careful navigation through the solution spaces, we have to keep track of a various number of possible solutions in order to compute a score attached to each possible solution. This score allows us to choose the most suitable one.

Roughly speaking, an alignment between two characters string `A` and `B` is a one to one correspondence of the characters of `A` with the characters of `B` or with the empty character denoted "_". The alignment process is symmetrical. Generally different possible alignments exist between two strings. Table 6 give three different possible alignments between A = **aaabbb** and B = **aaacbb**.

4.1 The Longest Common Subsequence Algorithm

The Longest Common Subsequence (LCS) algorithm is a well-known algorithm[3] used in string sequence comparison (seee for example [14,15]). The goal of this algorithm is to provide a longest common substring between two character strings.

More precisely, given a sequence $X = \langle x_1, x_2, ..., x_m \rangle$, another sequence $Z = \langle z_1, z_2, ..., z_n \rangle$ is a subsequence of X if there is a strictly increasing sequence of indices $\langle i_1, i_2, ... i_k \rangle$ such that $z_j = x_{i_j}$ for each $j \in [1 : k]$. For example, if $X = \langle A, B, C, D, A, B, C \rangle$ then $Z = \langle B, D, B, C \rangle$ is a subsequence of X. A common subsequence to sequences X and Y is a subsequence of both X and Y. Generally there is more than one LCS. We denote $|X|$ the length of X, and $X[i]$ the i^{th} character of that sequence.

Computing the LCS is equivalent to computing an edit distance between two character strings. An edit distance between sequences X and Y is the minimum number of operations such as suppression, addition and replacement (in term of characters) needed to change the sequence X into Y. An edit distance that is computed without the replacement operation is sometimes called *LCS distance* by some authors. This function is a kind of dual length of the length of an LCS between X and Y(see, for more details, Crochemore [16] Chap.7). The length of a LCS between X and Y will be denoted $lcs(X, Y)$ or simply lcs if there is no ambiguity. The edit distance and the LCS can be computed efficiently by the dynamic programming algorithm.

[3] The Unix `diff` command is based on this algorithm.

Table 7. Computation of an LCS matrix T

		y	a	m	i	m
	0	0	0	0	0	0
y	0	1	1	1	1	1
a	0	1	2	2	2	2
m	0	1	2	3	3	3
aa	0	1	2	3	3	3
m	0	1	2	3	3	4

Once the computation of an lcs is achieved, one can compute an alignment of the two sequences. Most of the time, one considers any of the alignments as equivalent. It will not be the case here, because the comparison is based on words, not only on characters.

Example 1. Let us compute the *lcs* between two (simple) Sanskrit texts: $X =$ `yamaan`, $Y =$ `yamin`. Note that according to the Velthuis transliteration `aa` is a single letter: long `a` (ā). The value of the lcs, here 4, is displayed at the bottom right corner of the matrix T. The distance between the two sequences is $d(X, Y) = |X| + |Y| - 2 * lcs(X, Y)$. In this exemple $d(X, Y) = 5 + 5 - 2 * 4 = 2$ (the letter `m` is suppressed and the letter `aa` is added).

The matrix is initialised to zero, and each score is computed by:

$$T[i, j] = \begin{cases} T[i-1, j-1] + 1 & \text{if } X[i] = Y[j], \\ max\{T[i-1, j], T[i, j-1]\} & \text{otherwise.} \end{cases}$$

The score $T[i, j]$ gives the value of the lcs between subsequences $X[1 : i]$ (the i first characters of the sequence X) and $Y[1 : j]$. These subsequences are defined as the first i letters of X and j letters of Y respectively. Each score $T[i, j]$ can be computed using some adjacent scores as shown in the previous formula. The complexity of the matrix computation is obviously in $O(|X||Y|)$. In this example, the LCS matrix generates exactly the two following symmetrical alignments.

Table 8. The two possible alignments

y	a	m	i	_	m
y	a	m	_	aa	m

y	a	m	_	i	m
y	a	m	aa	_	m

The alignment can be read in the following way: when letters are present in the same column of the two rows, they belong to the LCS. When a letter `l` is present with an opposite “-”, then `l` can be considered either as added in the line where it appears, or suppressed from the line where the opposite “-” is present.

Example 2. The comparison between two short sentences, as shown in Figure 1, describes the way we proceed and what kind of result can be expected. The sentences compared in this example are:
tasmai śrīgurave namas and *śrīgaṇeśāya namaḥ*, which are encoded:

`tasmai "srii_gurave namas` and `"sriiga.ne"saaya nama.h`

		"s	r	ii	g	a	.n	e	"s	aa	y	a	n	a	m	a	.h
t	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
a	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
s	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
m	0	0	0	0	0	1	1	1	1	1	1	1	1	1	2	2	2
ai	0	0	0	0	0	1	1	1	1	1	1	1	1	1	2	2	2
"s	0	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2
r	0	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
ii	0	1	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3
g	0	1	2	3	4	4	4	4	4	4	4	4	4	4	4	4	4
u	0	1	2	3	4	4	4	4	4	4	4	4	4	4	4	4	4
r	0	1	2	3	4	4	4	4	4	4	4	4	4	4	4	4	4
a	0	1	2	3	4	5	5	5	5	5	5	5	5	5	5	5	5
v	0	1	2	3	4	5	5	5	5	5	5	5	5	5	5	5	5
e	0	1	2	3	4	5	5	6	6	6	6	6	6	6	6	6	6
n	0	1	2	3	4	5	5	6	6	6	6	6	7	7	7	7	7
a	0	1	2	3	4	5	5	6	6	6	6	7	7	8	8	8	8
m	0	1	2	3	4	5	5	6	6	6	6	7	7	8	9	9	9
a	0	1	2	3	4	5	5	6	6	6	6	7	7	8	9	10	10
.h	0	1	2	3	4	5	5	6	6	6	6	7	7	8	9	10	11

Fig. 1. A second example

Note that the first sentence (X) belongs to the *padapāṭha*, the second (Y) to a *mātṛkāpāṭha*, and that the character "_" (underscore) is a lemmatization sign.
The matrix in Figure 1 contains all the possible alignments, one of them being the alignment in Table 9. We can see that the string `tasmai` is missing in the *mātṛkāpāṭha*, that the string `"srii` is present in both sentences, that `gurave` is replaced by `ga.ne"saaya`, and that the string `nama.h` is present in both sentences but under two different aspects: "`nama.h`" and "`namas`". The rule that states the equivalence between character `.h` and character `s` is one of the *sandhi*'s (see: 3.2). The following alignment is one of the possible results, the separation between words of the *padapāṭha* being represented by double vertical lines.

We can see in this example that the value of the *lcs*(X, Y) is 14 and it appears in the right bottom corner of the table. The distance between X and Y expressed in terms of letters is:

$$d(X,Y) = |X| + |Y| - 2 * lcs(X,Y) = 16 + 19 - 2 * 14 = 7$$

In terms of words, one word is missing: `tasmai`; the word `gurave` can be considered as replaced by `ga.ne"saaya` or missing in the *padapāṭha* and `ga.ne"saaya` added in the *mātṛkāpāṭha*. The value of the distance in terms of words will be either two or three according to the definition of the replacement operation.

During our comparison process, we must keep in mind that our final goal is to provide a difference between a *mātṛkāpāṭha* and the *padapāṭha* in terms of words. To appreciate the quality of this difference, an implicit criterion is to say that **the fewer words concerned, the better the criterion**, all things being equal, the word boundaries being provided by the *padapāṭha*.

Table 9. The corresponding alignment

t	a	s	m	ai	"s	r	ii	g	u	r	a	_	v	e	_	_	_	_	n	a	m	a	s
_	_	_	_	_	"s	r	ii	g	_	_	a	.n	_	e	"s	aa	y	a	n	a	m	a	.h

Table 10. different comparisons

1c1	1d0	Word 1 'tasmai' is :
< "sriigane"saayanama.h	< tasmai	- Missing
---	4c3,5	Word 2 '"srii' is :
> tasmai"sriiguravenama.h	< gurave	- Followed by
	---	Added word(s)
	> gane	'ga.ne"saaya'
	> "	Word 3 'gurave' is :
	> saaya	- Missing
diff without space	ediff with space	Our results without space

Consequently, in what follows we will choose, whenever possible, the solution which not only minimises the number of words concerned, but also, as far as no other criteria are involved, minimises the number of letters concerned.

4.2 Why Not Use the `Diff` Algorithm

The authors very first idea was to use `diff` in order to obtain the differences between two sanskrit sequences. It is stated in `diff` documentation that the inspiration of the actual version of `diff` was provided by the paper of Myers ([17])

But the results were quite disapointing. The classical `diff` command line provided no useful information at all. The result of the comparison of the two following sequences: `"srii ga.ne"saaya nama.h` and `tasmai "srii␣gurave namas` just said that they were different.

We obtained a slightly better result with `Emacs ediff`, as shown in Table 10, middle column: we can see which words are different. But as soon as we wanted to compare the same sequences without blank, we could not get a better result using `ediff` than using `diff`. This is why we started to implement our own algorithm. Its results appear in the right column of Table 10. We can see that they are expressed in term of words.

- Concerning `diff` and Myers's paper and all the derivated litterature, the emphasis is lain on the performance, for time as well as for space.
- Concerning our algorithm, no optimization has been applied, the main goal is to use the *padapāṭha* as a template on a *mātṛkāpāṭha* to determine, as well as possible, the end of words. Once we have determined the one to one correspondance between the words of the *mātṛkāpāṭha* and of the *padapāṭha*, we are nearly finished and there only remains to compare two Sanskrit letter strings to see their differences. Obviously, the added or missing words have to be noted carefully.

4.3 Sailing through the LCS Matrix

The LCS matrix is only a base for further computations. What we need is an alignment which can provide us with some reasonable results. Each alignment corresponds to a path within the matrix. A short explanation of the construction of an alignment can be found in the first chapter of [18] or in [16]

The matrix provides alignments coming from the rightmost lowest corner to the leftmost upper corner (inverse order from the usual reading direction) in the following way:

```
Original Text                          : vaidikaanaa.m laukikaanaa.m

Text of File jntest.txt                : laukikaanaa.m vaidikaanaa.m

         |  ||  ||a||  ||  ||  ||a||  ||a||.||  ||a||  ||  ||  ||a||  ||a||.|
         |  ||l||u||k||i||k||a||n||a||m||v||i||d||i||k||a||n||a||m|

          0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
v   |     0  0  0  0  0  0  0  0  0  0  1  1  1  1  1  1  1  1  1
ai  |     0  0  0  0  0  0  0  0  0  0  1  2  2  2  2  2  2  2  2
d   |     0  0  0  0  0  0  0  0  0  0  1  2  3  3  3  3  3  3  3
i   |     0  0  0  0  1  1  1  1  1  1  1  2  3  4  4  4  4  4  4
k   |     0  0  0  1  1  2  2  2  2  2  2  2  3  4  5  5  5  5  5
aa  |     0  0  0  1  1  2  3  3  3  3  3  3  3  4  5  6  6  6  6
n   |     0  0  0  1  1  2  3  4  4  4  4  4  4  4  5  6  7  7  7
aa  |     0  0  0  1  1  2  3  4  5  5  5  5  5  5  5  6  7  8  8
.m  |     0  0  0  1  1  2  3  4  5  6  6  6  6  6  6  6  7  8  9
l   |     0  1  1  1  1  2  3  4  5  6  6  6  6  6  6  6  7  8  9
au  |     0  1  2  2  2  2  3  4  5  6  6  6  6  6  6  6  7  8  9
k   |     0  1  2  3  3  3  3  4  5  6  6  6  6  6  7  7  7  8  9
i   |     0  1  2  3  4  4  4  4  5  6  6  6  6  7  7  7  7  8  9
k   |     0  1  2  3  4  5  5  5  5  6  6  6  6  7  8  8  8  8  9
aa  |     0  1  2  3  4  5  6  6  6  6  6  6  6  7  8  9  9  9  9
n   |     0  1  2  3  4  5  6  7  7  7  7  7  7  7  8  9 10 10 10
aa  |     0  1  2  3  4  5  6  7  8  8  8  8  8  8  8  9 10 11 11
.m  |     0  1  2  3  4  5  6  7  8  9  9  9  9  9  9  9 10 11 12
```

Fig. 2. The different alignments within the matrix

1. if $\boldsymbol{T}[\boldsymbol{i}, \boldsymbol{j}] < \boldsymbol{T}[\boldsymbol{i}+\mathbf{1}, \boldsymbol{j}+\mathbf{1}]$ and if $\boldsymbol{X}[\boldsymbol{i}] = \boldsymbol{Y}[\boldsymbol{j}]$ we move (left and up) from $\boldsymbol{T}[\boldsymbol{i}+\mathbf{1}, \boldsymbol{j}+\mathbf{1}]$ to $\boldsymbol{T}[\boldsymbol{i}, \boldsymbol{j}]$ and in this case, the score, which is decreased by 1, indicates that a (common) letter has been added to the left of the alignment. $\boldsymbol{A} = \begin{pmatrix} \boldsymbol{X}[\boldsymbol{i}] \\ \boldsymbol{Y}[\boldsymbol{j}] \end{pmatrix} .\boldsymbol{A}$ (the dot indicates the concatenation operation).
2. otherwise, if $\boldsymbol{T}[\boldsymbol{i}, \boldsymbol{j}] < \boldsymbol{T}[\boldsymbol{i}, \boldsymbol{j}+\mathbf{1}]$ we move vertically up one row and add $\begin{pmatrix} \boldsymbol{X} \\ - \end{pmatrix}$ at the beginning of the alignment A = $\begin{pmatrix} \boldsymbol{X} \\ - \end{pmatrix} .\boldsymbol{A}$.
3. otherwise, we move horizontally one column left. In this case, add $\begin{pmatrix} - \\ \boldsymbol{X} \end{pmatrix}$ to the alignment.

Figure 2 (p. 372) presents all the alignments provided by the LCS algorithm in an LCS matrix. The dark grey line depicts the chosen alignment, and the light grey lines represent other alignments also provided by the LCS algorithm. The sequence $\boldsymbol{X}$ belonging to the *padapāṭha*, the alignments are selected in order to maximise the number of consecutive letters belonging to $\boldsymbol{X}$. This choice reduces the risk for two parts of the same word in the *padapāṭha* to be identified with two different subsequences of the *mātṛkāpāṭha*.

The chosen alignment corresponding to the dark grey line is depicted in Table 11.

It may be pointed out that when the different paths through the matrix form a square (no common letters can be found between $\boldsymbol{X}$ and $\boldsymbol{Y}$ at this place), the number of possible alignments grows very quickly. If $\boldsymbol{N}$ is the size of the square, the number of different alignments generated by each square is:

$$\binom{2N}{N} = \frac{(2N)!}{N!N!}$$

To provide a good idea of the possible number of paths, if we have a matrix which contains two ten by ten squares we get approximately 39×10^9 different possible alignments. This number expresses how complicated the comparison of Sanskrit texts is, and excludes any method that would require to examine all the possible alignments produced by the LCS algorithm.

Table 11. The chosen alignment

v	ai	d	i	_	_	_	_	k	aa	n	aa	.m	l	au	k	_	_	_	i	k	aa	n	aa	.m
_	_	_	_	l	au	k	i	k	aa	n	aa	.m	_	_	_	v	ai	d	i	k	aa	n	aa	.m

4.4 Optimization: Some Navigation Rules

In order to restrict the number of alignments, we will provide some navigation rules within the matrix. These navigation rules will greatly limit the number of solutions to be considered but they are unable to provide a good solution by themselves. Other steps are necessary to obtain a solution which gives some satisfaction to the philologists.

Let us try to give an idea of the different navigation rules implemented within the program. They concern the best way to choose a path (corresponding to an alignment) in the LCS matrix. Though in the preceding paragraph we described, for mathematical reason, the navigation through the matrix in the upward direction, right to left, we will now describe this navigation in the usual order, downward, and left right, which is easier to understand.

As a first remark we must notice that when the different paths form a square which corresponds to a place where there is no letter in common between the strings X and Y, we always go down first as in Fig. 2. It induces to write in the alignment the part of the *padapāṭha* corresponding the square first, then write the corresponding part of the *mātṛkāpāṭha*.

But the great question we will always keep in mind during the navigation through the matrix is: **shall we align the soonest or the latest sequence**? The answer to this question will determine the navigation rules[4].

Table 12 shows two examples, the left one needs to be aligned **the latest** in order to provide the good result, on the contrary, the right one needs to be aligned **the soonest**. In each figure, the right path will be displayed in dark grey, the wrong one in light grey.

– On the left example, we see the comparison between two strings, in the *mātṛkāpāṭha*: `a~n` and in the *padapāṭha*: `a.n a~n`. The LCS matrix is displayed and the corresponding alignments just under. The good solution in the table is the best according to common sense, it is also the best according to our criterion: **the fewer words concerned, the better the criterion**. The conclusion of this alignment is: the string `.n` is missing in the *mātṛkāpāṭha*.

[4] Our examples are sometimes taken from Sanskrit manuscripts, sometimes built for demonstration purpose, without any Sanskrit meaning.

Table 12.

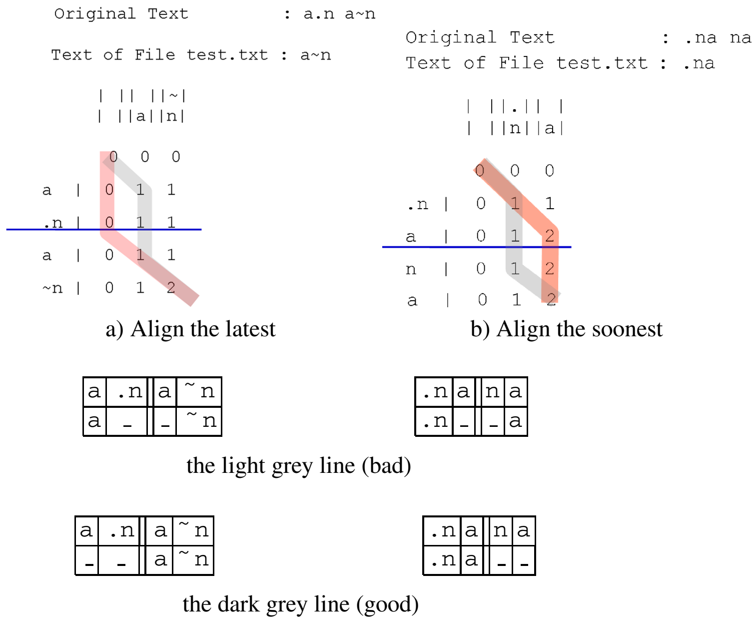


Table 13.

```
Original Text :avi
Manuscript    :bhaviavi

  |  ||b || ||  ||  || || || |
  |  ||h||a||v||i||a||v||i||

        0  0  0  0  0  0  0  0
a  |    0  0  1  1  1  1  1  1
v  |    0  0  1  2  2  2  2  2
i  |    0  0  1  2  3  3  3  3
```

a) rule 2

-	-	-	-	a	v	i
bh	a	v	i	a	v	i

Align the latest.

```
Original Text:avi
Manuscript   :avibhavi

 |  || || ||  ||h || || || |
 |  ||a||v||i||b||a||v||i||

        0  0  0  0  0  0  0  0
a  |    0  1  1  1  1  1  1  1
v  |    0  1  2  2  2  2  2  2
i  |    0  1  2  3  3  3  3  3
```

b) rule 3

a	v	i				
a	v	i	bh	a	v	i

Align the soonest.

– On the right example we see the comparison between two strings, in the *mātṛkāpāṭha*: `.na` and in the *padapāṭha*: `.na na`. The LCS Matrix is displayed with its corresponding alignments. The conclusion of this alignment is: the string `na` is missing in the *mātṛkāpāṭha*.

Our second example, in Table 13 is different with a *mātṛkāpāṭha* involving more letters than the *padapāṭha*. Only the **good** alignments are displayed.

What kind of conclusion can we draw from these apparently contradictory samples? This set of rules:

1. By default align the latest.
2. If, while aligning the soonest, we cross one of the *padapāṭha* word boundaries, then align the soonest.
3. If the choice occurs at the end of a *padapāṭha* word, then align the latest without further checking.
4. If, while aligning the soonest, we cross one of the *padapāṭha* word boundaries, then align the soonest.

The limit of words which are determined by the *padapāṭha* are the major determinant of the navigation rules. The rules displayed here are not complete, others exist (not described here), but they are more or less based on the same principles.

4.5 Improvement of the Initial LCS Alignment by the Use of a Score

As the first author of this paper has absolutely no knowledge of Sanskrit, he was looking for evaluating this result, and he found that our first criterion **if fewer words are concerned, the criterion is better** must be followed by another one: the **compactness** of the alignment. The following example provides an idea of what we expect:

.r	k	aa	r	e	e	v	a	a	c	k	aa	r	y	aa	.n	i
.r	-	aa	r	-	-	-	-	-	-	-	-	-	y	aa	.n	i

The alignment has been built according to the navigation rules. It can be interpreted as: the word *kāre* is replaced by *ār*, the words *eva* and *ac* are missing, the word *kāryāṇi* is replaced by *yāṇi*.

.r	k	aa	r	e	e	v	a	a	c	k	aa	r	y	aa	.n	i
.r	-	-	-	-	-	-	-	-	-	-	aa	r	y	aa	.n	i

The second alignment is built taking compactness into account, it can be interpreted as: the words *kāre*, *eva* and *ac* are missing, the word *kāryāṇi* is replaced by *āryāṇi*, (the letter k is missing), which is obviously the best solution.

4.6 Problems Which Cannot Be Solved by the LCS

The identification of words in the *mātṛkāpāṭha*, as implicitly defined from the previous alignments, is not completely satisfactory. Indeed the maximisation of the lcs cannot fulfill our purpose, because the value of the lcs is only related to the notion of character, whereas our aim is to compare the texts word by word. Once the alignment is obtained, the words of the *mātṛkāpāṭha* are not really identified. To improve this alignment we propose a procedure which consists in local changes of the alignment to fulfill the following two rules:

1. Two words cannot be considered as similar if they do not share at least 50% of their characters (very short words must be considered apart).

2. Considering that words can be suppressed, added, or replaced in the *mātṛkāpāṭha*, the desired alignment has to minimise the number of those operations.

Notice that the second rule matches exactly the definition of the edit distance, but in terms of words instead of characters as is usually the case. The results provided by these two rules were approved by the philologists in charge of the Sanskrit critical edition. To illustrate our approach let us compare the following two texts: `upadi"syate mahaa .n` in the *padapāṭha* and a *mātṛkāpāṭha* with: `upadi.syata.n`. The LCS algorithm provides an alignment with an ***lcs*** of 10 that does not fulfill rule number 1.

u	p	a	d	i	"s	_	y	a	t	e	m	a	h	aa	.n
u	p	a	d	i	_	.s	y	a	t	_	_	a	_	_	.n

This involves the following conclusions:

– The string `upadi"syate` is replaced by `upadi.syat`
– The word `mahaa` is replaced by `a`

The next alignment is not optimal for the LCS criterion, because its ***lcs*** is only 9, but is preferable because the first rule is satisfied:

u	p	a	d	i	"s	_	y	a	t	e	_	m	a	h	aa	.n
u	p	a	d	i	_	.s	y	a	t	_	a	_	_	_	_	.n

– the string `upadi"syate` is replaced by `upadi.syata`
– the string `mahaa` is missing

It appears that the improvement of the initial alignment consists in asserting that the string `mahaa` is missing instead of stating that the string `maha` is replaced by `a`.

4.7 Pending Problems

There are two major lacks in the software:

– If a long text is added to the *mātṛkāpāṭha* we are unable to see what are the words that compose it, because the *padapāṭha* is useless in this case.
– More important, we missed an important point at the beginning of the software conception: if a word is changed or is missing in a text, most probably *sandhi* will be changed. But the *sandhi* rules are applied at the beginning of the process, during the transformation of the *padapāṭha* into the *saṃhitapāṭha*, so we may have, in some cases, to reconsider the *sandhis* at the end of the process.

5 Displaying the Result

The results of the comparison program are first displayed as a `log` file as it was the best way for the necessary program tuning.

```
Paragraph 3 is Missing in File Asb2

(P3) Word 6 'paaraaya.na' is:
 - Substituted with 'paaraya.naa' in Manuscript ba2

(P3) Word 11 'saara' is:
 - Substituted with 'saadhu' in Manuscript aa

(P3) Word 17 'viv.rta' is:
 - Followed by Added word(s) 'grantha"saa' in Manuscript A3

(P3) Word 18 'guu.dha' is:
 - Missing in Manuscript A3

(P3) Word 21 'viudpanna' is:
 - Substituted with 'vyutpannaa' in Manuscript A3

(P3) Words 22 to 23 'ruupa siddhis' are:
 - Missing in Manuscript A3

(P3) Word 32 'k.rtyam' is:
 - Substituted with 'karyam' in Manuscript A3
 - Substituted with 'kaaryam' in Manuscripts aa, am4, ba2
```

After a conversion of these logged information into XML language, from which we can obtain a HTML file which can provide us an interactive version of the critical edition. Figure 3 gives an example of such a display.

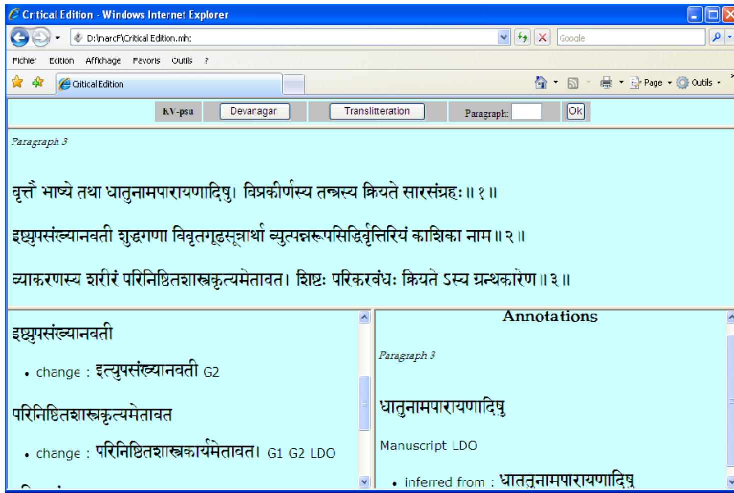


Fig. 3. Example of interactive display of the results

6 Conclusion

In this paper we have proposed a method to compare different versions of the same Sanskrit text. The alignments provided by the LCS algorithm between two texts, considered as a sequence of characters, is not always sufficient, but provides a good initialisation for further processing that considers each of the two texts as sequences of words.

The critical edition provided by such improved alignments has been submitted to philologists and has been approved in its essential part. Nevertheless a more intense use of the software should enable us to improve and justify the setting of our empirical approach. There is also a serious need to completely rewrite the software to avoid the different dead end procedures which are still present and make the program maintenance too complicated. We also need to make more experiments for a better tuning.

The program works at a reasonable speed. With a *padapāṭha* of approximately 300 lines, and 45 different *mātṛkāpāṭha* the time needed for the comparison process is approximately 25 seconds. It seems to be quite reasonable.

However, the absence of a Sanskrit lexicon constitutes a limit to our approach: in the case of an addition of long sentences to a manuscript, it is impossible to detect words which have been added, for we can only consider the addition in terms of sequence of characters.

References

1. O'Hara, R.J., Robinson, P.: Computer-assisted methods of stemmatic analysis. In: Blake, N., Robinson, P. (eds.) Occasional Papers of the Canterbury Tales Project. Office for Humanities Communication, vol. 1, pp. 53–74. Office for Humanities Communication, Oxford Univerity (1993)
2. Monroy, C., et al.: Visualization of variants in textual collations to analyse the evolution of literary works in the cervantes project. In: Agosti, M., Thanos, C. (eds.) ECDL 2002. LNCS, vol. 2458, pp. 638–653. Springer, Heidelberg (2002)
3. Münster University: Digital nestle-aland (2006), http://nestlealand.uni-muenster.de/index.html
4. Monier-Williams, M.: A Sanskrit English Dictionary. Clarendon Press, Oxford (1899)
5. Velthuis, F.: Devanāgarī for TEX, Version 1.2, User Manual (1991), http://www.ctan.org/tex-archive/language/devanagari/velthuis/
6. Le Pouliquen, M.: Filiation de manuscrits sanskrit et arbres phylogénétiques. In: Mathématiques & Sciences Humaines (accepted for publication) (2007)
7. Gale, W.A., Church, K.: A program for aligning sentences in bilingual corpora. Computational Linguistics 19(3), 75–102 (1993)
8. Saitou, N., Nei, M.: The neighbour-joining method: a new method for reconstructing phylogenetic trees. Molecular Biology Evolution 4, 406–425 (1987)
9. Huet, G.: Héritage du sanskrit: Dictionnaire français-sanskrit (2006), http://sanskrit.inria.fr/Dico.pd
10. Huet, G.: Design of a lexical database for sanskrit. In: COLING Workshop on Electronic Dictionaries, Geneva, pp. 8–14 (2004)
11. Paxson, V.: GNU Flex Manual, Version 2.5.3. Free Software Foundation, Cambridge, Mass. (1996), http://www.gnu.org/software/flex/manual/
12. Lesk, M.E., Schmidt, E.: Computing science technical report 39. Technical report, Bell Laboratories. Murray Hill, NJ (1975)

13. Renou, L.: Grammaire sanskrite: phonétique, composition, dérivation, le nom, le verbe, la phrase. Maisonneuve, Paris (1996) (réimpression)
14. Hunt, J.W., Szymanski, T.: Fast algorithm for computing longest common subsequence. CACM 20(5), 350–353 (1977)
15. Hirschberg, D.: A linear space algorithm for computing maximal common subsequences. CACM 18, 341–343 (1975)
16. Crochemore, M., Hancart, C., Lecroq, T.: Algorithms on Strings. Cambridge University Press, Cambridge (2007)
17. Myers, E.W.: An O(ND) difference algorithm and its variations. Algorithmica 1(2), 251–266 (1986)
18. Charras, C., Lecroq, T.: Sequence comparaison, `http://www-igm.univ-mlv.fr/~lecroq/seqcomp/seqcomp.ps`
19. Haralambous, Y.: Fonts & Encodings. O'reilly Media, Inc., Sebastopol (2007)

Chi-Squares and the Phenomenon of "Change of Exemplar" in the *Dyūtaparvan*

Wendy J. Phillips-Rodriguez*, Christopher J. Howe, and Heather F. Windram

Institute for Philological Research, National Autonomous University of Mexico, 04510, Mexico City
Department of Biochemistry, University of Cambridge, Tennis Court Road, Cambridge CB2 1QW, UK
wjp26@cantab.net, ch26@mole.bio.cam.ac.uk, H.F.Windram@btinternet.com

Abstract. In this paper we investigate the use of computational tools in the textual analysis of Sanskrit manuscripts. We use the maximum chi-squared method, a technique borrowed from molecular biology, to analyse the distribution of variants in two different pairs of manuscripts of the *Dyūtaparvan*. This method gives some insight into the process of manuscript copying in the *Mahābhārata* tradition. In particular it provides evidence that at least one of the scribes used more than one exemplar to produce his own version of the *Mahābhārata*.

Keywords: textual analysis, stemmatics, contamination, chi-squares, change of exemplar.

1 Introduction

In the last few decades there has been an increasing awareness that computational tools can be very helpful in the textual analysis and editing of ancient texts that survive in a number of witnesses. From the initial stage of cataloguing and collating manuscripts through to the final publication of an edition, there are several intermediate steps where the use of computers can prove extremely helpful; for example, in the textual analysis required to establish the relationships between the extant manuscripts.

More recently, several interdisciplinary projects have used techniques developed in the field of evolutionary biology to map the genealogical relationships between manuscripts [Barbrook *et al.* 1998, Howe *et al.* 2001]. These studies have made use of phylogenetic algorithms based on the principles of cladistics and other evolutionary methods originally created for the purpose of mapping the relationships between species from their DNA sequences. As Macé & Baret [2006: 90] affirm: "The basic hypothesis that makes this common work possible is that the diversity of the stages of a text in a manuscript tradition and the differences between biological taxa can be explained by the evolution of this text or taxa from a single origin".

These phylogenetic algorithms have achieved considerable success in mapping vertical relationships between manuscripts. However, less attention has been paid to mapping horizontal relationships, where the use of more than one exemplar gives rise to a contaminated text.

G. Huet, A. Kulkarni, and P. Scharf (Eds.): Sanskrit CL 2007/2008, LNAI 5402, pp. 380–390, 2009.

Contamination can be regarded in simple terms as falling into two categories, simultaneous and successive. The former is where a scribe used two (or more) exemplar copies at the same time to make a copy, with readings from each other interspersed. Successive contamination occurs when a scribe is copying from one manuscript, but at some point he changes exemplar and starts copying from another manuscript for a substantial part (or parts) or for the remainder of the text, and thus there may be lengthy sections of text that come from a different source. Contamination can be very difficult to trace, with simultaneous contamination proving particularly difficult for statistical analysis. Successive contamination is more accessible, and we used the maximum chi-squared method [Windram *et al.* 2005] to investigate possible exemplar change within one section of the *Mahābhārata* tradition.

2 Methods

Computer-based methods were used to analyse data from the *Mahābhārata* tradition. Variant information was obtained for sargas 43-47, 51, 59-61, and 64-65 of the *Dyūtaparvan*, which appears in the *Sabhāparvan*, the second book of the *Mahābhārata*. Not all of these sargas are consecutive, as we decided to use only the ones that contain the most significant narrative facts. We focused on the 31 manuscripts collated in the Poona Critical Edition [Sukthankar, 1933] and added the Critical Edition (CE) itself to see how the reconstructed text relates to the other witnesses of the tradition.

Encoding of Text

The data for the selected sargas were represented as a numerical matrix. The variants were encoded using a numerical system, such that at locations where two manuscripts have an identical variant they are given an identical number and where two manuscripts differ they are given different numbers. In this way a matrix, or NEXUS file [Maddison et al, 1977], was created with each row of the matrix representing a manuscript and each column, or character, representing a variant location (usually a word) in the text. The NEXUS file for the *Mahābhārata* data consisted of a 32 manuscript x 5235 character matrix.

Phylogenetic Analysis of All Manuscripts

The data matrix was analyzed using the Neighbor-joining algorithm of Saitou and Nei [1987]. This method creates a distance matrix based on the number of differences between manuscripts and uses this information to create a tree that best represents the distances between all manuscripts. The branch lengths of the trees are proportional to the number of differences between texts. Trees were also generated using shorter sections of the data in order to identify any manuscripts that may change their affiliation throughout the text.

Bootstrap analysis [Felsenstein, 1985] was used to give a measure of statistical support for the tree structure generated by Neighbor-joining. Bootstrapping is a resampling technique which uses a large number of replicate data sets, generated from the original data set by a random sampling method. A phylogenetic tree is generated for each data set and the results are represented as a consensus tree which only shows manuscript groupings supported by at least 50% of the individual trees (all figures

shown in this paper are bootstrap consensus trees). The bootstrap support value gives a measure of the statistical support for a particular grouping. As a guide, in biological systems, a group is considered to be well supported if it has a bootstrap support value of 70% or over [Hillis and Bull, 1993]. Important bootstrap values are given at the appropriate places in the text.

Analysis of Pairs of Manuscripts

Data from selected pairs of manuscripts were used for further analysis.

Maximum Chi-squared Analysis

This method is based on the chi-squared analysis developed for the detection and mapping of the genetic equivalent of contamination, recombination between sequences of biological data [Maynard Smith, 1992]. Using the data from the NEXUS file for a selected pair of manuscripts, the program creates a sequence showing the locations where there are differences between the two texts. If the two texts are similarly related throughout, then it would be expected that the differences between the texts would, on average, be fairly evenly distributed along the sequence. However, the observed pattern of differences may change suddenly if one of the texts shifts to another exemplar. The program moves an imaginary breakpoint along the sequence of differences, and compares the observed number of differences before and after this breakpoint with the expected number of differences in the absence of any exemplar change or recombination, and calculates a chi-squared value for each breakpoint. The chi-squared values are plotted against the location of the breakpoint. The maximum chi-squared value, and the highest peak of the chart, occurs at the location where there is the greatest discrepancy between the observed and expected distribution of differences, and where a change of exemplar is most likely to have occurred [Windram *et al.*, 2005].

Percentage of Differences Plot

A window of specified size is moved along the sequence of differences created for the chi-squared analysis. The number of locations where there is a difference between the two texts is calculated as a percentage of the total number of characters in the window. This value is plotted at the midpoint character number (the window size isalways an odd number of characters) and the window is then moved on by one character. A window size of 501 characters proved optimal for the *Mahābhārata* data. This analysis indicates the direction of change on each side of a breakpoint identified using the maximum chi-squared method, i.e. whether texts become more or less similar.

3 Results

The complete NEXUS file (32mss x 5235chr) prepared for the selected sargas was analyzed using the Neighbor-joining algorithm as implemented by PAUP* [Swofford, 2001] to give an indication of the relationships between the manuscripts in this region of the *Mahābhārata* (Fig. 1).

The tree shown is unrooted, as Neighbor-joining makes no assumption about the origin of the tradition. It may help to think of it as a tree seen from an aerial

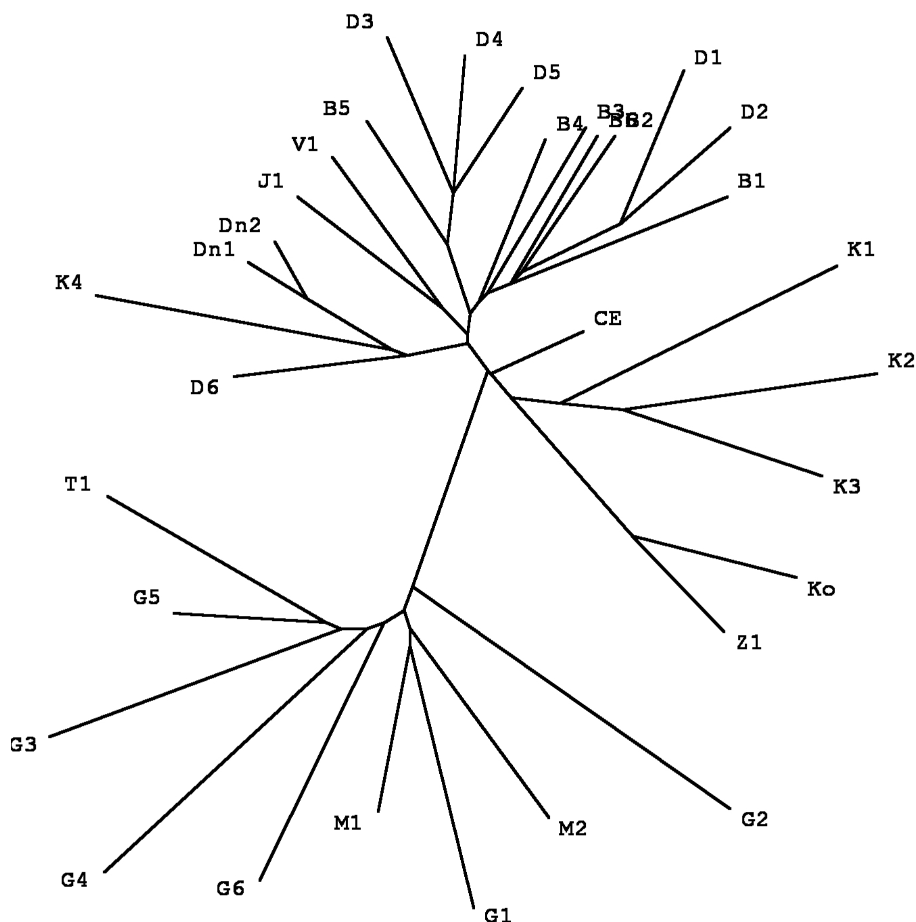


Fig. 1. Critical Edition=CE. *Eastern group*: D=Devanāgarī, B=Bengali, J=Nepālī, V=Maithilī, Dn=Vulgate. *Kashmirian group*: Z=Śāradā, K=D mss of Z. *Southern group*: T=Telugu, G=Grantha, M=Malayālam. Unrooted Neighbor-joining tree of ten sargas of the *Dyūtaparvan* in the thirty-one manuscripts collated for the Critical Edition, and the Critical Edition itself.

perspective, where it is only possible to see the branches and how are they related to each other, but the root of the tree remains out of sight. The manuscripts divide into three groups. The major division occurs between the Northern and the Southern families of manuscripts (100% bootstrap support). The Southern group consists of all the G, M and T manuscripts written in the Grantha, Malayālam and Telugu scripts respectively. However, the Northern recension clearly shows a split into two further groups (100% support): the Kashmirian group consisting of both the Śāradā manuscript (Z) and the Devanāgarī versions of the Śāradā text (the K manuscripts), and the Northeastern or Eastern family containing the Devanāgarī (D), Bengali (B), Nepali (J) and Maithili (V) manuscripts and the Devanāgarī versions of the Vulgate text (Dn manuscripts). Within this Northeastern family, the B and D manuscripts form a separate

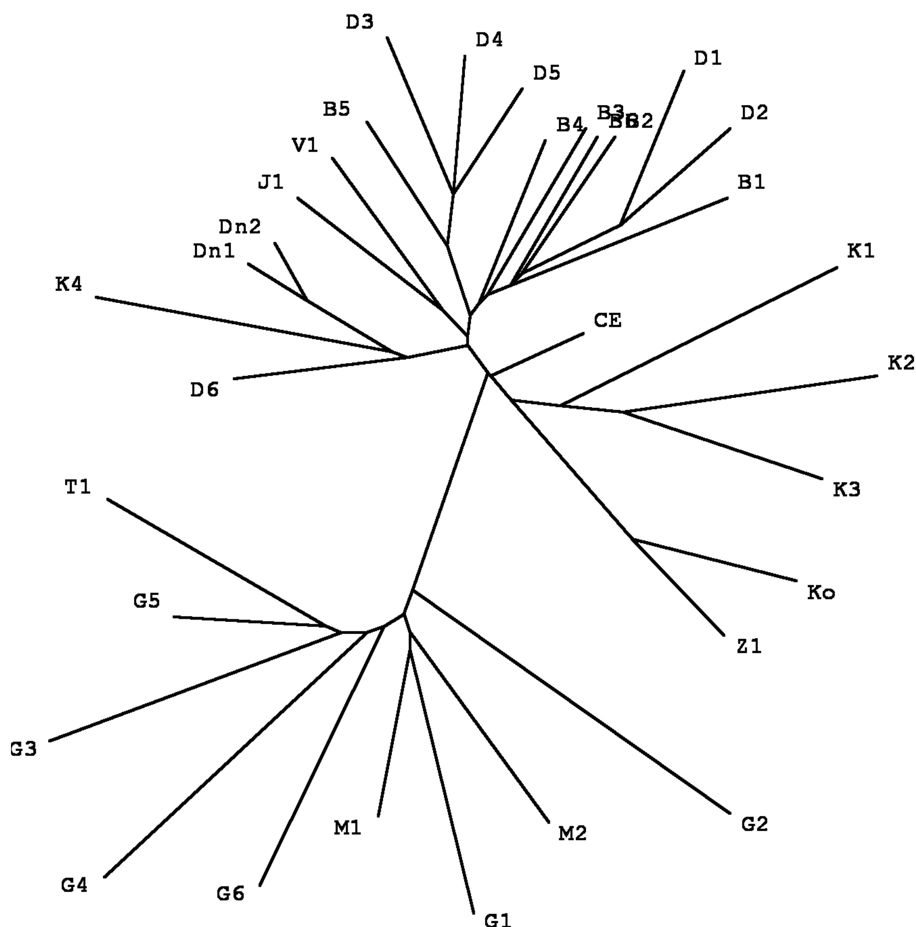


Fig. 2. Critical Edition=CE. *Eastern group*: D=Devanāgarī, B=Bengali, J=Nepālī, V=Maithilī, Dn=Vulgate. *Kashmirian group*: Z=Śāradā, K=D mss of Z. *Southern group*: T=Telugu, G=Grantha, M=Malayālam. Unrooted Neighbor-joining tree of section 1 (chr 1-1745).

subgroup (BD) (100% support), in which the manuscripts are not grouping according to their Bengali or Devanāgarī script. Manuscript K4 seems out of place in the tree, grouping with the Vulgate manuscripts Dn1 and Dn2 (89% support), but according to the evidence presented by Sukthankar and Edgerton this manuscript is a hybrid between a Kashmirian and a Vulgate version.[1] In this particular part of the *Dyūtaparvan*, K4 may be closer to its Vulgate than to its Kashmirian source. Manuscript D6 also

[1] In his Prolegomena, Sukthankar´s description of K4 (which he in fact calls D1) says that it was ¨copied from different exemplars; some parvans have the commentary of NīlakaıÒha, while others contain some old text tradition¨ (1933: XVIII). In the second volume of the edition, Edgerton added that the agreements of K4 with other manuscripts of the tradition "seem to me to make it quite certain that the 'old text tradition' noted by Sukthankar as blended in the manuscript with a vulgate source, was no other than the Kashmirian" (1944: XII).

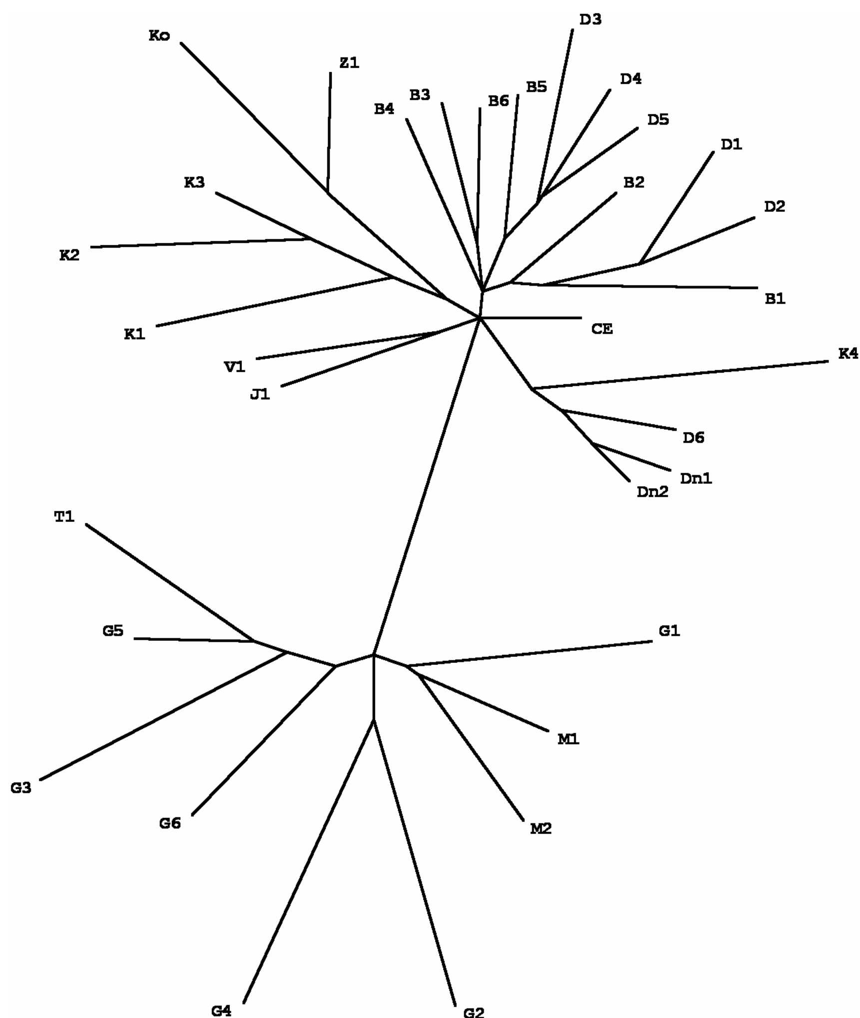


Fig. 3. Critical Edition=CE. *Eastern group*: D=Devanāgarī, B=Bengali, J=Nepālī, V=Maithilī, Dn=Vulgate. *Kashmirian group*: Z=Śāradā, K=D mss of Z. *Southern group*: T=Telugu, G=Grantha, M=Malayālam. Unrooted Neighbor-joining tree of section 2 (chr 1746-3490).

groups (100% support) with the Vulgate versions and with K4 rather than with the other D manuscripts. The text of the Critical Edition (CE) appears to be most closely related to the Northern recension.

In a tradition of this size there is a large number (32 x 31/2 = 496) of possible manuscript pairs. In order to select the most suitable manuscripts for pairwise analysis, the NEXUS file was then split into three sections, each of 1745 chr, and Neighbor-joining trees prepared for each section to give an indication of any manuscripts that changed their affiliation within the ten selected sargas of the *Dyūtaparvan*.

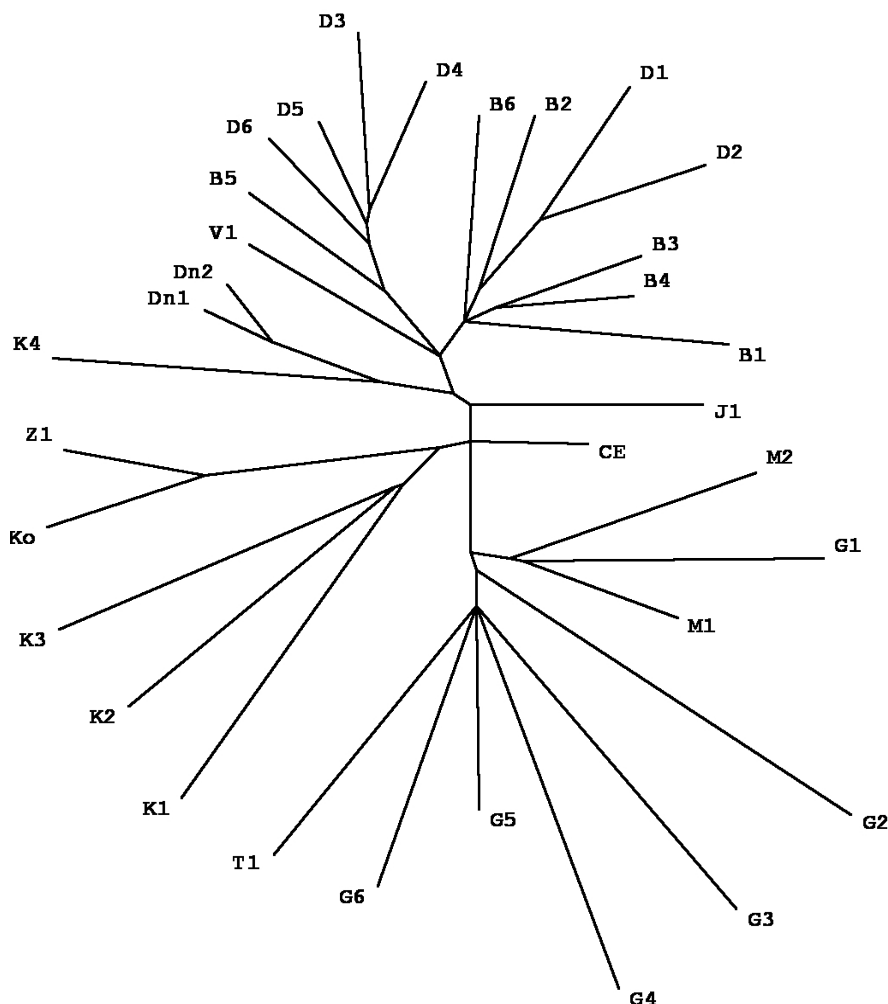


Fig. 4. Critical Edition=CE. *Eastern group*: D=Devanāgarī, B=Bengali, J=Nepālī, V=Maithilī, Dn=Vulgate. *Kashmirian group*: Z=Śāradā, K=D mss of Z. *Southern group*: T=Telugu, G=Grantha, M=Malayālam. Unrooted. Neighbor-joining tree of section 3 (chr 3491-5235)

The Neighbor-joining trees for section 1 (Fig. 2) and section 2 (Fig. 3) show little change in the manuscript groupings from that seen in the complete tradition, with both manuscripts K4 and D6 grouping with the Vulgate texts (99% support in section 1; 100% support in section 2). However, in the tree for section 3 (Fig. 4), manuscript D6 shifts its affiliation to join the other members of the BD subgroup (grouping with B5, D5, D3 and D4 with 100% support), while K4 remains with the Dn manuscripts (100% support).

This affiliation shift of D6 indicated that it might be a suitable candidate for pair-wise analysis along with its closest neighbours both before (Dn1) and after (D5) the

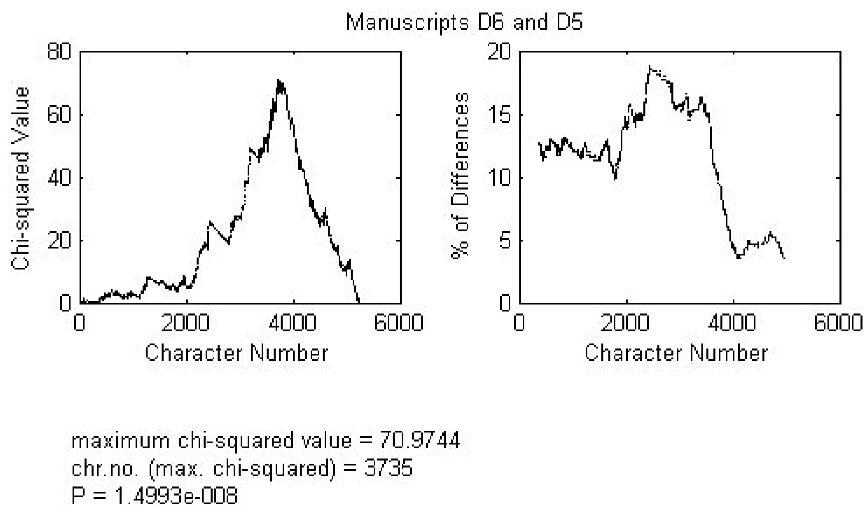


Fig. 5. Plot of chi-squared values (left) and the percentage of differences (right) against the location in the text for the manuscripts D6 and D5. The chart (right) shows the percentage of differences in the region of coded text contained in a moving window (of 501 characters) against the character number at the mid-point of the window.

shift. Maximum chi-squared and percentage difference analyses were performed for manuscript pairs D6/Dn1 and D6/D5.

As D6 showed a clear shift of affiliation, it was decided to perform the analysis in a way that would enable the statistical significance of the result to be determined. The significance threshold corresponding to a conventional false-positive probability of 0.05 may be determined for 32 manuscripts (31 manuscripts + the CE text) where the test manuscript is compared with its nearest neighbour on either side of a possible breakpoint. The significance threshold is determined by the formula for the Dunn-Sidak method, as described in Sokal and Rohlf [1995: 239]. A probability value (P) is determined for each pairwise comparison. This indicates the likelihood of obtaining this result in the absence of a genuine breakpoint. Any P value higher than the significance threshold is statistically insignificant.

$$\text{Significance threshold} = 1 - 0.95^{(1/64)} = 8.014 \times 10^{-4} \; (= 0.0008 \text{ approx}) . \qquad (1)$$

In the chi-squared plot for the manuscript pair D6/D5 (Fig. 5) the highest peak occurs at character 3735, indicating that, statistically, this is the most likely location for a breakpoint e.g. a change of exemplar. The probability value $P = 1.5 \times 10^{-8}$, which is statistically significant as it falls well beneath the threshold value. The percentage of differences plot indicates that the two manuscripts become much more similar in a region that corresponds with the location of the maximum chi-squared value. These results taken together would be consistent with manuscript D6 suddenly moving closer to manuscript D5 in this region.

The results for the D6/Dn1 pairwise comparisons are shown in Figure 6. The maximum chi-squared peak occurs at character 3757, which corresponds to the region

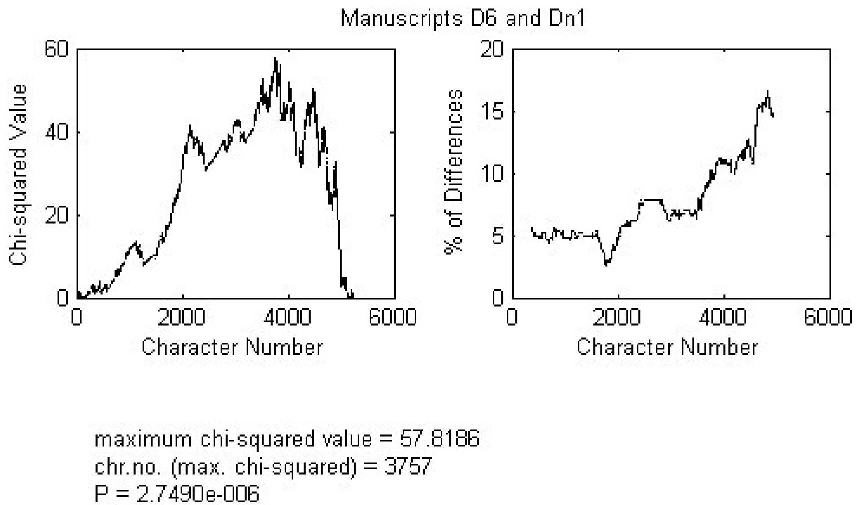


Fig. 6. Plot of chi-squared values (left) and the percentage of differences (right) against the location in the text for the manuscripts D6 and Dn1. The chart (right) shows the percentage of differences in the region of coded text contained in a moving window (of 501 characters) against the character number at the mid-point of the window.

where the percentage of differences plot indicates that the two manuscripts start to become more different from each other. A probability value of 2.7 x 10^{-6} indicates that the results are statistically significant.

The results of the D6/D5 and the D6/Dn1 comparisons are complementary and together clearly show the shift in affiliation of manuscript D6 away from Dn1 and towards D5. The analyses indicate that this shift occurs between characters 3735 and 3757. These two values are in very good agreement, and indicate that a change of exemplar may have occurred after the end of sarga 51, whose last word is located at character 3747.

However, no data for chapters 52 to 58 were included in the analyses, and sarga 51 was followed by sargas 59 and 60 in the NEXUS file.[2] Thus, if any change of exemplar occurred, it could have occurred within the region between the end of sarga 51 and the beginning of sarga 59. Work is currently in progress to prepare a complete and detailed analysis of the whole *Dyūtaparvan*.

[2] In terms of plot, it is important to remember what occurs in sargas 59 and 60. 59: Duryodhana orders Vidura to fetch Draupadī, he refuses with dark predictions (ślokas 1-10). 60: Duryodhana orders an usher to fetch her; he goes and tells Draupadī (1-5). Draupadī tells him to ask in the assembly if Yudhi˘Òhira lost himself before losing her; Yudhi˘Òhira gives no answer (5). Duryodhana orders the usher to bring Draupadī (10). Draupadī appears in the hall, clothed in one garment (15). Duryodhana tells Du˛śāsana to fetch Draupadī; he goes and drags her by the hair into the hall (15-20). She protests that she has her period (25-30). She appears; Du˛śāsana insults her (35). Bhī˘ma fails to solve the problem whether she was Yudhi˘Òhira's to stake (40). Draupadī protests (40-45). Description taken from J. A. B. van Buitenen (1975: 106-109).

4 Discussion

The analyses indicate that there is a change in the relationship between manuscript D6 and the Vulgate (Dn) and BD texts in the *Dyūtaparvan*. The Dn manuscripts are known to contain contaminated texts because we know that NīlakaıÒha, when preparing the Vulgate text in the last quarter of the seventeenth century, “had compared many copies of the *Mahābhārata*, collected *from different parts of India*, with a view to determining the ‘best’ readings and even consulted the scholia of old authorities” [Sukthankar, 1933: LXV].[3] The Vulgate manuscripts are not all dated, but they can scarcely be earlier than the beginning of the eighteenth century. Manuscript D6 has been dated 1570 AD, and therefore it predates the Vulgate manuscripts.

Since D6 is an earlier manuscript than any of the Vulgate texts, it is clearly impossible for the D6 scribe to have switched from a Vulgate to a BD-type exemplar. Instead we have a more complex situation where there are a variety of possible scenarios:

- D6 may be just a copy of another manuscript where the change of exemplar may have actually taken place, from an exemplar similar to the one used by NīlakaıÒha in the first two analyzed sections of the *Dyūtaparvan* to a BD-type one.
- The scribe of D6 himself could have made the change of exemplar
- The scribe of the Vulgate (e.g. NīlakaıÒha) may have shifted from a D6-type exemplar to other(s) or started introducing a great number of readings that are not available in D6.[4]

These are by no means all the possibilities, just a few of them to illustrate that the change of exemplar could have occurred before D6, in D6 or after D6. The motivations for such change of exemplar also remain unknown. The scribe may have made a conscious choice by picking the text he wanted to copy or he may have been forced to do so, e.g. if the manuscript he was copying from was incomplete.

This means that, according to our evidence, there has been at least one case of successive contamination during the Mahabharata manuscript transmission. However, it is not possible to know exactly when it occurred. D6 is our only indicator of this change of exemplar but being a single manuscript it throws very little light into what actually happened.

This confirms the need to increase the number of manuscripts analyzed. The Critical Edition of this part of the text was prepared using data from thirty-one manuscripts, which was a large number to deal with by manual methods. However, with the availability of modern computer based methods for the preparation and analysis of textual data, it should be possible to extend the data base to include a far greater number of the extant manuscripts. The availability of a bigger pool of variant readings should facilitate the production of more varied and complete editions. In addition, it may also offer some insight into the processes of textual evolution that were at work during the development of the *Mahābhārata* manuscript tradition.

---

[3] In the quotation cursives are ours.

[4] However, the results as they stand at the moment cannot really be interpreted solely as the Vulgate manuscripts moving away from D6. This would explain why D6 moves away from Dn1 but not why D6 moves towards D5.

Acknowledgements

The textual analyses were performed as part of the TEXTNET project, funded by the Leverhulme Trust. We are grateful to Barbara Bordalejo and Peter Robinson for many helpful discussions and suggestions.

References

Barbrook, A.C., Howe, C.J., Blake, N., Robinson, P.: The phylogeny of the Canterbury Tales. Nature 394, 839 (1998)

van Buitenen, J.A.B. (trans.): The Mahābhārata: 2 The Book of the Assembly Hall & 3 The Book of the Forest. The University of Chicago Press (1975)

Edgerton, F.D. (crit. ed.): Sabhāparvan. In: Sukthankar, V.S., et al. (crit. eds.) The Mahābhārata, vol. II. Bhandarkar Oriental Research Institute, Poona (1944)

Felsenstein, J.: Confidence limits on phylogenies: an approach using the bootstrap. Evolution 39, 783–791 (1985)

Hillis, D.M., Bull, J.J.: An empirical test of bootstrapping as a method for assessing confidence in phylogenetic analysis. Systematic Biology 42, 182–192 (1993)

Howe, C.J., Barbrook, A.C., Spencer, M., Robinson, P., Bordalejo, B., Mooney, L.R.: Manuscript evolution. Endeavour 25, 121–126 (2001)

Macé, C., Baret, P.: Why phylogenetic methods work: the theory of evolution and textual criticism. In: Macé, C., Baret, P., Bozzi, A., Cignoni, L. (eds.) Linguistica Computazionale. The evolution of texts: confronting stemmatological and genetical methods, vol. XXIV-XXV, pp. 89–108. Istituti Editoriali e Poligrafici Internazionali, Pisa-Roma (2006)

Maddison, D.R., Swofford, D.L., Maddison, W.P.: NEXUS: An extensible file format for systematic information. Systematic Biology 46, 590–621 (1997)

Mahdavi, A.M.: Genetically Modified Text' or 'Critical Edition'? In: The Shāhnāma Genome Project. Persica, vol. XIX, pp. 1–31. Peeters, Leuven (2003)

Maynard Smith, J.: Analysing the Mosaic Structure of Genes. Journal of Molecular Evolution 34, 126–129 (1992)

Windram, H.F., Howe, C.J., Spencer, M.: The identification of exemplar change in the Wife of Bath's Prologue using the maximum chi-squared method. Literary and Linguistic Computing 20, 189–204 (2005)

Windram, H.F., Spencer, M., Howe, C.J.: Phylogenetic analysis of manuscript traditions, and the problem of contamination. In: Macé, C., Baret, P., Bozzi, A., Cignoni, L. (eds.) Linguistica Computazionale. The evolution of texts: confronting stemmatological and genetical methods, vol. XXIV-XXV, pp. 141–156. Istituti Editoriali e Poligrafici Internazionali, Pisa-Roma (2006)

Spencer, M., Davidson, E.A., Barbrook, A.C., Howe, C.J.: Phylogenetics of artificial manuscripts. Journal of Theoretical Biology 227, 503–511 (2004)

Saitou, N., Nei, M.: The Neighbor-joining Method: A New Method for Reconstructing Phylogenetic Trees. Molecular Biology and Evolution 4, 406–425 (1987)

Sokal, R.R., Rohlf, F.J.: Biometry. W.H. Freeman and Co., New York (1995)

Sukthankar, V.S.: Âdiparvan. In: The Mahābhārata, vol. I. Bhandarkar Oriental Research Institute, Poona (1933)

Swofford, D.L.: PAUP*. Phylogenetic Analysis Using Parsimony (*and other methods). Sinauer Associates, Sunderland (2001)

Applying the OCRopus OCR System to Scholarly Sanskrit Literature

Thomas M. Breuel

DFKI and University of Kaiserslautern
Kaiserslautern, Germany
tmb@informatik.uni-kl.de

Abstract. OCRopus is an open source OCR system currently being developed, intended to be omni-lingual and omni-script. In addition to modern digital library applications, applications of the system include capturing and recognizing classical literature, as well as the large body of research literature about classics. OCRopus advances the state of the art in a number of ways, including the ability easily to plug in new text recognition and layout analysis modules, the use of adaptive and user extensible character recognition, and statistical and trainable layout analysis. Of particular interest for computational linguistics applications is the consistent use of probability estimates throughout the system and the use of weighted finite state transducers to represent both alternative recognition hypotheses and statistical language models. In this paper, I first give an overview of these technologies and their relevance to digital library applications in the humanities, and then focus on the use of statistical language models and their use for the integration of OCR output with subsequent computational linguistic and information extraction modules.

1 Introduction

New digitization techniques, cheap storage, cheap imaging equipment, and fast networking are making large amounts of scholarly literature available in image format, including literature in, and about, Sanskrit. Unfortunately, much of this literature remains inaccessible as existing OCR systems cannot cope well with this type of content, for a variety of reasons.

Large scale scanning efforts, like Google Books and various Million Books projects, tend to rely on scanning methods that result in fairly low image quality relative to the kinds of scanning methods that commercial OCR systems tend to be developed for.

Recognition of Devanagari has been studied, and some academic and commercial systems have been developed, but their performance is not yet at the level of systems for English. Some of the reasons are the greater complexity of the Devanagari script, the large numbers of ligatures, and the large and unusual vocabulary used in academic and historical texts. In addition, scholarly and classical uses of Devanagari may use special typographic features not found in commercial uses of Devanagari.

G. Huet, A. Kulkarni, and P. Scharf (Eds.): Sanskrit CL 2007/2008, LNAI 5402, pp. 391–402, 2009.

In addition, scholarly texts are frequently multi-lingual and multi-script, mixing Devanagari and Latin script, and occasionally using Greek letters and IPA symbols.

Overall, for classics applications, we can at least distinguish the following document types:

– *original documents.* Some documents of scholarly relevance are indeed the original written records–the first fixation of words into written form by the original authors; these may be handwritten, severely degraded, and use unusual fonts and styles.
– *original texts.* Many classical texts have been recorded, edited, and reprinted within the last several centuries, using modern printing techniques and modern typography. Such texts are of great scholarly interest, as are detailed comparisons between the different editions. Although the body of such books primarily consists of the original text, they often also contain scholarly commentary, marginal notes, footnotes, and annotations, requiring multi-script and multi-language OCR.
– *commentaries, analyses, textbooks.* These kinds of documents contain a mix of scholarly writing in modern languages, passages in classical languages, annotations, footnotes, references, and other content.
– *dictionaries, encyclopedias, catalogs.* These kinds of documents contain large amounts of structured information. They are often characterized by non-standard grammars, complex layouts, and complex mixes of scripts and languages.

The OCRopus system (Breuel, 2008) is a multi-lingual and multi-script open source document analysis and OCR system that is actively being developed. In this paper, I will review some of the directions we are taking in adapting OCRopus to the needs of digitizing and analyzing scholarly literature, with an emphasis on Sanskrit.

2 The OCRopus System

Unlike many commercial OCR systems, the OCRopus system uses a strictly feed-forward architecture (Figure 1). That is, each processing stage receives some input and computes an output representation that is then used by the next stage, with no backtracking. From a software engineering point of view, this greatly reduces coupling between components and makes it much easier to "plug in" other layout analysis and text recognition modules. From a pattern recognition point of view, it is not known how to model backtracking correctly statistically, making it difficult to evaluate, test, or optimize OCR systems based on backtracking.

The major stages of the OCRopus system are:

– *Preprocessing* consists of page dewarping and deskewing, noise removal, and page frame detection.

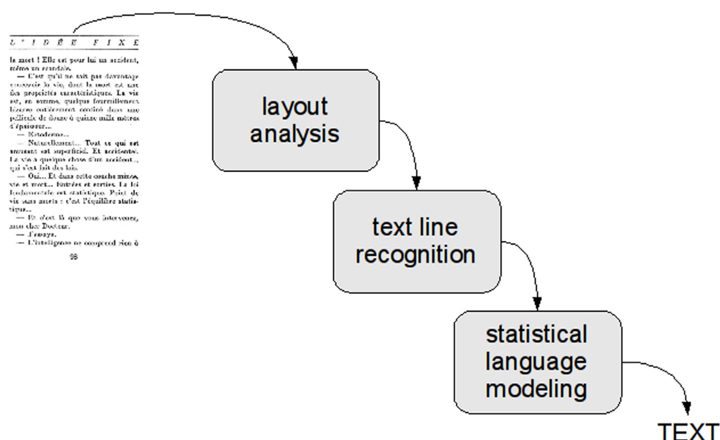


Fig. 1. The OCRopus system architecture is strictly feed-forward, with the three major stages being layout analysis, text line recognition, and statistical language modeling

- *Layout analysis* performs a separation of text and images on a page, and divides the text regions into text columns, paragraphs and lines.
- *Text line recognition* recognizes linear sequences of characters and outputs the corresponding text and style information, in a format that allows for both segmentation and recognition alternatives.
- *Statistical language modeling* selects the best interpretation of the text taking into account prior knowledge about the vocabulary, orthography, and grammar of the target language.

OCRopus uses only a small number of data types internally, making it easy to reuse parts of OCRopus in other systems, and making it easy to incorporate other systems into OCRopus:

- Images are used for representing input data, as well as page layouts, segmentations, and other spatial data structures.
- Weighted finite state transducers are used for representing the output of text line recognition (that is, text with associated segmentation and recognition alternatives), as well as statistical language models.
- The hOCR format is used for representing the final output of the OCR system; it encodes OCR information in completely standards-compliant HTML files. By building upon existing HTML standards, hOCR automatically has a rich set of representations for typographic phenomena in the world's major languages.

OCRopus is primarily a set of C++ libraries implementing a variety of different document analysis and OCR-related functions. OCRopus contains a set of simple standard internal C++ abstract interfaces for major document analysis and OCR functionality, like isolated character recognition, text line recognition,

```
while pages:nextPage() do
   pages:getBinary(page_image)
   segmenter:segment(page_segmentation,page_image)
   regions = RegionExtractor()
   regions:setPageLines(page_segmentation)
   page = PageNode()
   for i = 1,regions:length()-1 do
      regions:extract(line_image,page_image,i,1)
      lattice = make_FstBuilder()
      bpnet_recognizer:recognizeLine(lattice,line_image)
      bestpath2(line,lattice,langmod)
      page:append(line:utf8())
   end
   document:append(page)
end
```

Fig. 2. The top-level driver loop (in Lua) for multi-page, multi-column OCR with statistical language modeling. Users can easily modify this loop, for example, to use a different recognizer, to change the way language modeling is handled, to construct a different output format, or to pre-process pages or lines to remove noise.

segmentation, and layout analysis. In addition, most of the OCRopus functionality is available through scripting languages, and major functions are available as command line programs (Figure 2).

OCRopus can be used in a variety of ways for document analysis and OCR:

- Users can access it through hosted web services, submitting images through a web browser or command line web client, and obtaining document analysis and OCR results as web pages or images.
- Users can run OCRopus from the command line or use it from shell scripts.
- Users can customize how OCRopus performs OCR by scripting the OCR engine in Lua (a widely used, easy-to-use scripting language); in fact, the top-level driver loops are all written in Lua and can be modified to suit particular needs (e.g., different image preprocessing, different output formats).
- Users can add C++ functions and components to OCRopus. Components that conform to the existing OCR-related interfaces can be used as drop-in replacements for functions like layout analysis and text line recognition. Pre-existing OCR systems can be integrated into OCRopus in this way.

3 Preprocessing

The OCRopus OCR system has a fully scriptable library of efficient image processing and statistical operations, including efficient binary morphology and geometric transformations. Obvious uses of these tools include noise removal and other document cleanup.

However, many standard document analysis algorithms can also be formulated as image processing tasks. For example, layout analysis by smearing, layout

analysis using the docstrum, and similar high-level operations can be expressed in terms of binary morphology, distance transforms, marker morphology, and statistics.

Even simple shape recognition problems, such as finding staffs, bars, circles, and lines can be expressed using these primitives.

4 Layout Analysis

OCRopus features a number of different layout analysis methods:

– The primary layout analysis method used in OCRopus is based on computational geometry algorithms for finding whitespace and robust least square matching.
– We are developing a trainable statistical layout analysis method that will be included in future versions of OCRopus.
– XY cuts and Voronoi methods are widely used methods from the literature. In benchmarks, we find that these methods do not genernally perform as well as the previous two methods, but they may be useful in some specialized applications.
– Custom "smearing" and morphologically based layout analysis methods can be explicitly scripted in Lua and used in place of one of the built-in methods.

Two key questions that arise in the context of applications to scholarly Sanskrit literature are how well these algorithms work on Devanagari, and how well they work on mixed text.

The primary layout method used in OCRopus performs layout analysis in several steps:

– First, it computes a whitespace cover of the background, by computing all locally maximal rectangles.
– From this whitespace cover, it selects column separators.
– Then it performs robust least square matching of text lines against the connected components on the page, subject to the constraint that text lines cannot cross column separators.
– Finally, it performs reading order determination by computing a "text-line-after" relationship between pairs of text lines and extending that partial order to a total order via topological sorting.

Generally speaking, OCRopus's layout analysis methods seem to work quite well on both types of input (see Figure 3). Text column styling is similar between the two languages and there is no need to modify the white space cover algorithm. The fact that Devanagari largely lacks intra-word whitespace actually reduces the probability of column misdetection.

The constrained text line finder in OCRopus uses a "western style" text line model based on baselines and descenders; however, this model works well for

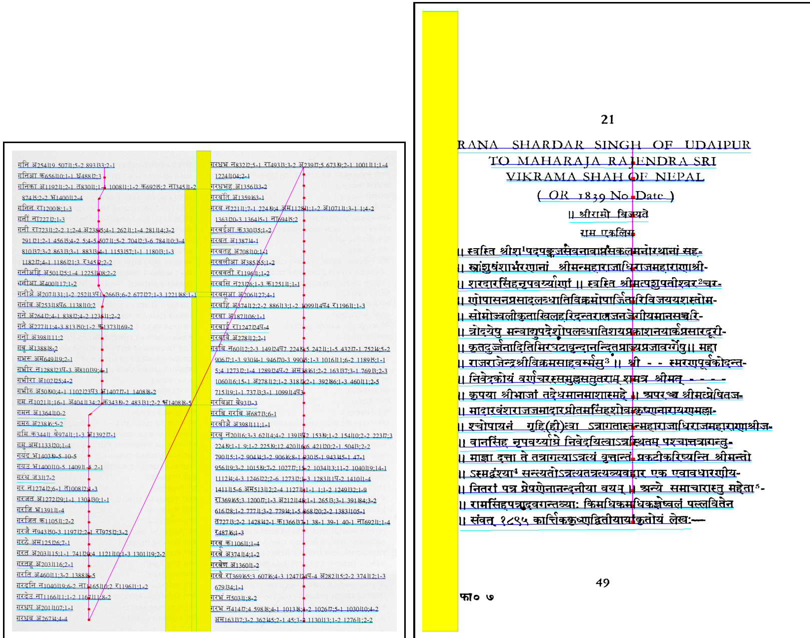


Fig. 3. The result of applying the standard OCRopus page layout analysis tool to documents consisting of mixed Devanagari and Latin script. Standard OCRopus line and column finders work well on Devanagari text, as well as mixed Devanagari/Latin text.

Devanagari, since Devanagari characters effectively have size variations corresponding to descenders and ascenders. However, the statistics for those variations are somewhat different from those of Latin script, with a larger "x-height" relative to the size of the descenders and ascenders.

Layout analysis also appears to work well on mixed Latin/Devanagari text using the current model (see Figure 3). However, in the medium term, it will be desirable to modify OCRopus text line finding to perform a pre-classification into Devanagari and Roman characters and to create separate text line models for the two types of text. The prominent horizontal line running through most Devanagari characters may provide a convenient, if ad hoc solution to this classification problem until a more general purpose, trainable pre-classification mechanism is implemented in OCRopus.

5 Line Recognizers

Although "OCR for classical Sanskrit" might primarily be understood as recognition of Sanskrit texts written in Devanagari, the task is actually considerably broader. Some historical texts may use different writing systems, since Devanagari is not the only script in historical use for Sanskrit. Scholarly writing on

Sanskrit almost always uses Latin script, and Latin script is also used for writing Sanskrit itself, including extended passages. Sanskrit written in Devanagari and Latin scripts also makes use of numerous diacritics that need to be recognized. In addition, IPA may be used for pronunciation, Greek letters may be used for classical Greek quotations, and Greek letters and other special characters may be used for indicating footnotes or other references. Text recognition for literature involving Sanskrit therefore needs to be able to cope with this wide variety of symbols, as well as mixing them within the same text and even the same line or sentence.

As indicated above, OCRopus abstracts all actual character recognition in the form of text line recognizers; that is, text line recognizers for different scripts are responsible for segmenting the text line into characters, then recognizing the characters, and then returning a data structure that represents all possible segmentation- and recognition alternatives for the input line. This approach allows the same interface to be used between the rest of the system and the character recognition engine despite the fact that different scripts require different approaches to word and character segmentation.

5.1 Built-In Line Recognizers

OCRopus already has built-in line recognizers for Latin scripts. These are important in their own right in OCR systems for recognizing scholarly texts on Sanskrit, since, in order to recognize Latin script texts, for scholarly texts in languages like English, German, or French, and for recognizing Romanized Devanagari, an OCR system needs a high performance, high accuracy text line recognizer. By the beta release, OCRopus will have four such recognizers: (1) the Tesseract recognizer, a shape matching recognizer based on character outlines (Smith, 2007), (2) a multilayer perceptron (MLP)-based recognizer that performs oversegmentation followed by character classification with an MLP, (3) a shape-based recognizer using a hierarchical database organization and robust shape matching under transformations, and (4) a Hidden Markov Model (HMM) recognizer that integrates segmentation and recognition. By the 1.0 release, OCRopus will likely also incorporate a text line recognizer based on word shape.

Unlike many other approaches to OCR, these recognizers make few assumptions about the character set or fonts they are recognizing; instead, they are trained on text line input, perform alignment against ground truth data, and then train individual character shape models automatically. The feature set used by these recognizers are generic and applicable to many scripts. Language modeling is handled separately (see below). Recognition accuracy of these recognizers approaches that of commercial OCR systems.

Diacritics are currently handled by treating character+diacritic combinations as novel characters--an approach that works well for the limited character+ diacritic combinations found in modern European languages. Scholarly literature has a much large number of possible combinations, for example when diacritics are used to indicate stress. In addition, some diacritics in scholarly literature

span multiple characters. Increased use of OCRopus for scholarly literature may therefore require the development of diacritic recognition and removal prior to character recognition, with an integration of character recognition results and diacritics at a later stage.

5.2 External Line Recognizers

As already mentioned above, OCRopus makes it easy to incorporate external OCR and character recognition engines into the system. This will permit the use of Devanagari OCR engines that have already been developed within OCRopus for text line recognition, while still taking advantage of the layout analysis and statistical language processing facilities of OCRopus.

5.3 Devanagari

It will be desirable to take advantage of the built-in text line recognition methods in OCRopus for Devanagari recognition. Although the visual appearance of Devanagari is very different, its basic approach to writing is not all that different from Latin scripts (in fact, it has been hypothesized that the two writing systems share a common origin): Devanagari consists of a linear sequence of graphemes. Each grapheme represents a consonant or vowel, although consonants carry an implicit vowel. Phenomena like diacritics, word spacing, and kerning are common to both writing systems.

There are some significant differences to Latin script:

- In printed Latin script, graphemes do not touch each other in clean renditions of the script; in Devanagari, touching characters are the norm.
- The text line model is different, with character “hanging from” a horizontal line running through most of the Devanagari text.
- In Latin script, grapheme order corresponds mostly to phoneme order, while in Devanagari, order is frequently violated. For example, graphemes representing vowels frequently precede graphemes representing consonants, even though the vowel phoneme follows the consonant. Unicode encoding follows phoneme order in many cases, necessitating a transformation of grapheme to Unicode order.
- Devanagari makes extensive use of ligatures for consonant clusters; this means that although the basic character set is about the same size as Latin character sets, there is a much larger set of ligatures that needs to be recognized.
- Devanagari makes extensive use of diacritics; as in Latin script, these can either be recognized as part of the character itself, resulting in a large character set, or they can be recognized as separate units and later be combined with the base character set.

Many of these issues also arise with other writing systems and therefore need to be addressed within a multi-script, multi-lingual system. For example, Arabic and Urdu both are mostly-touching writing systems and have different text

line models from Latin script. And Urdu and historic Latin script both make extensive use of ligatures and diacritics. CJK languages (Chinese, Japanese, and Korean) require recognizers and classifiers that can cope with character sets containing tens of thousands of characters.

Our general approach to direct recognition of Devanagari within OCRopus is to adapt the existing recognizers. The two recognizers most easily adapted are the HMM recognizer and the MLP recognizer.

In order to enable training, the first step that will be necessary will be to decide on a set of graphemes for representing Devanagari text lines. Some of the graphemes will be treated as separate characters, while others may be treated as diacritics. Then, we need to construct an invertible Unicode-to-grapheme mapping; the mapping can be represented and computed out using finite state transducers, the same technology used for language modeling within the OCRopus system.

For the HMM recognizer, models are trained using standard methods: the forward-backward algorithm, the construction of HMM models for each character, and forced alignment with ground truth. Alignment has to be carried out against the grapheme sequence, not the original Unicode character sequence.

An oversegmenting MLP-based recognizer can likewise be trained directly against the grapheme sequence (where it is important that the graphemes chosen correspond to the "characters" identified by the segmenter).

In Figure 4, we see that the primary segmenter used within OCRopus for Latin scripts, a segmenter based on computing near-vertical cuts using a dynamic programming algorithm (Breuel, 2001), also yields reasonable grapheme sequences for Devanagari; the output of this segmenter can then be used as input for MLP-based grapheme recognition, and the resulting hypothesis graph can then be transformed into a unicode character sequences using the grapheme-to-Unicode mapping established previously.

A second segmenter within OCRopus is based on skeletal features and leads to a much higher degree of oversegmentation; it could potentially be used with other sets of graphemes.

6 Language Modeling

As already mentioned above, OCRopus relies for both the representation of segmentation and recognition alternatives, as well as for the representation of statistical language models, on *weighted finite state transducers* (Mohri *et al., 2002* WFSTs). WFSTs allow language models and recognition alternatives to be manipulated algebraically: they can be concatenated, unioned, intersected, composed, minimized, reversed, complemented, and transformed in a variety of other ways.

WFSTs can be thought of as directed graphs whose edges are associated with symbols and weights. The symbols are often Unicode characters, but can also represent graphemes, phonemes, ligatures, fonts and styles, and even entire strings.

Fig. 4. Segmentations of an input string (top) into character parts. Character parts are indicated by color. Character parts are grouped together into character hypotheses, which are then recognized by the character shape recognizer. Character parts that span multiple actual characters frequently lead to recognition errors (undersegmentation). On the other hand, splitting characters into many small parts results in longer recognition times (oversegmentation). Two segmentations are shown, obtained using standard OCRopus segmenters: the first segmentation is based on the curved cut segmenter (a dynamic programming method), the second segmentation is based on the skeletal segmenter. Here, segmenters were used with their default parameter settings (performance can likely be improved further by adapting parameter settings to Devanagari).

WFSTs are a mature technology used, for example, in speech recognition (Mohri et al., 2000), information extraction (e.g., Kramer et al., 2007) and statistical machine translation (e.g., Kumar and Byrne, 2003). WFSTs can be constructed manually or learned from training data.

WFSTs can be used to implement the kinds of language processing problems traditionally encountered in OCR systems:

- Dictionary-based language models are very common in OCR systems; dictionaries can be represented efficiently as WFSTs by first constructing a WFST whose paths correspond to individual words and whose weights correspond to logarithmic word frequencies, then minimizing that WFST using WFST minimization algorithms, and finally computing the closure of that WFST. The resulting WFST is similar to a trie data structure with additional compression of word suffixes.
- WFSTs can be used for training by substituting the recognition language model in the OCR system with a special language model that accepts only the actual transcription of the input text; the result of recognition with this special language model is an input that has been forcibly aligned to the ground truth. This alignment is then used to extract images for training HMM or MLP classifiers.

Beyond these traditional uses, there is a wide variety of other potential applications. For example, WFST-based language models allow us to solve the following problems:

- For source documents written in multiple languages (e.g., English and Sanskrit), we can take existing language models for English and Sanskrit and combine them. As part of the combination, we can train or specify the probable locations and frequencies of transitions between the two language models, corresponding to, for example, isolated foreign words within one language, or long quotations.

– We can represent and learn (from sample data) the correspondence and statistics of modern dictionary forms and classical spelling variants of words using WFSTs. The resulting WFST can be used in both directions: it can be used to map historical documents onto modern orthography, it can be used for aligning and comparing modern and historical texts, and it can be applied for using language models developed for one orthography in the OCR recognition of documents written using the other orthography.
– A WFSTs model can be used for transliterating Devanagari into Latin script, and the same model can then also potentially be used for the reverse direction.
– Often, ground truth data for OCR training is not perfectly aligned with the actual image data due to transcription errors, different source editions, etc. Instead of using the exact transcription of the ground truth text, we can transform it into a WFST that can be used instead of ground truth and is tolerant to misalignment, transcription errors, insertions, and deletions.
– By composing such language models, we can, for example, use a modern transliterated version of a text using modern orthography as the ground truth data for training OCRopus on a historical text printed in Devanagari.
– Some source documents, like dictionaries and catalogs, are described by small grammars. For example, a dictionary entry may have a head word in Sanskrit notated in Devanagari, followed by IPA notation in brackets, followed by a word class in italic Latin script, followed by a translation in Latin script alternating with usages examples in Sanskrit, concluding with a list of numerical references. This sequence of script, style, and language changes can be represented as a WFST, and the WFST can drive not only the recognition of each part of the entry, but also mark the logical function of each substring.
– Statistical machine translation systems tend to be quite sensitive to OCR errors. However, most OCR errors are actually not strictly speaking errors, but instead places where the OCR system itself recognized an ambiguity and was forced to make a decision. OCRopus represents segmentation and recognition alternatives as WFSTs. By using the WFSTs generated by OCRopus directly as input to a statistical machine translation system (e.g., Kumar and Byrne, 2003), the error rate of the overall system can potentially be greatly reduced. Similar considerations apply to other statistical language processing steps based on OCR output, such as named entity recognition, parts of speech tagging, and parsing.

7 Other Applications

The renewed interest in OCR systems was, in part, motivated by the widespread scanning activities by organizations like Google and the Internet Archive. These efforts tend to be large scale operations with expensive imaging and post-processing equipment.

Separately from the OCRopus effort, our research group is also developing portable, low-cost technology for document image capture applications, intended

to make the ability to capture historical and scholarly literature universally available, as well as to enable handheld machine translation for both students and researchers in the field.

We have also developed non-OCR-based image alignment for historical documents, permitting the automated comparison of low-quality historical documents, as well as the recovery of annotations and differences between editions.

8 Discussion

This paper has described on-going work in the development of a multi-lingual OCR system, and how the architecture of the system is supporting multiple languages, scripts, layouts, and language models.

As a community project, the quickest way of obtaining support for Sanskrit and Devanagari is through community support. The infrastructure and tools are already in place for this, and, as we have seen, image pre-processing, layout analysis, and linguistic post-processing are already in place for this and function well for Devanagari. Simple Devanagari support will likely appear this year, both based on the Tesseract recognizer and based on the OCRopus MLP recognizer.

References

[Breuel, 1994] Breuel, T.: Language Modeling for a Real-World Handwriting Recognition Task. In: The British Society for the Study of Artificial Intelligence and the Simulation of Behaviour, Workshop on Computational Linguistics for Speech and Handwriting Recognition, Leeds University, England (1994)

[Breuel, 2008] Breuel, T.: The OCRopus Open Source OCR System. In: Proceedings, The Society for Imaging Science and Technology (IS&T) and Society of Photographic Instrumentation Engineers (SPIE), 20th Annual Symposium 2008 (2008)

[Breuel, 2001] Breuel, T.: Segmentation of handwritten letter strings using a dynamic programming algorithm. In: Proc. 6th Int. Conf. on Document Analysis and Recognition (DAS) (2001)

[Kumar and Byrne, 2003] Kumar, S., Byrne, W.: A weighted finite state transducer implementation of the alignment template model for statistical machine translation. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (2003)

[Kramer et al., 2007] Kramer, M., Kaprykowsky, H., Keysers, D., Breuel, T.: Bibliographic Meta-Data Extraction Using Probabilistic Finite State Transducers. In: International Conference on Document Analysis and Recognition (2007)

[Mohri et al., 2002] Mohri, M., Pereira, F., Riley, M.: Weighted Finite-State Transducers in Speech Recognition. Computer Speech and Language (2002)

[Setlur et al., 2003] Setlur, S., Kompalli, S., Ramanaprasad, V., Govindaraju, V.: Creation of data resources and design of an evaluation test bed for Devanagari script recognition. In: Proceedings of 13th International Workshop on Research Issues in Data Engineering: Multi-lingual Information Management (2003)

[Smith, 2007] Smith, R.: An Overview of the Tesseract OCR Engine. In: Proc. 9th Int. Conf. on Document Analysis and Recognition (ICDAR) (2007)

Keyword Spotting Techniques for Sanskrit Documents

Anurag Bhardwaj, Srirangaraj Setlur, and Venu Govindaraju

Center for Unified Biometrics and Sensors
Department of Computer Science and Engineering
University at Buffalo, Amherst NY – 14228
{ab94,setlur,govind}@cubs.buffalo.edu

Abstract. With advances in the field of digitization of printed documents and several mass digitization projects underway, information retrieval and document search have emerged as key research areas. However, most of the current work in these areas is limited to English and a few oriental languages. The lack of efficient solutions for Indic scripts and languages such as Sanskrit has hampered information extraction from a large body of documents of cultural and historical importance. This chapter presents two relevant topics in this area. First, we describe the use of a script specific Keyword Spotting for Sanskrit documents that makes use of domain knowledge of the script. Second, we address the needs of a digital library to provide access to a collection of documents from multiple scripts. This requires intelligent solutions which scale across different scripts. We present a script independent Keyword Spotting approach for this purpose. Experimental results illustrate the efficacy of our methods.

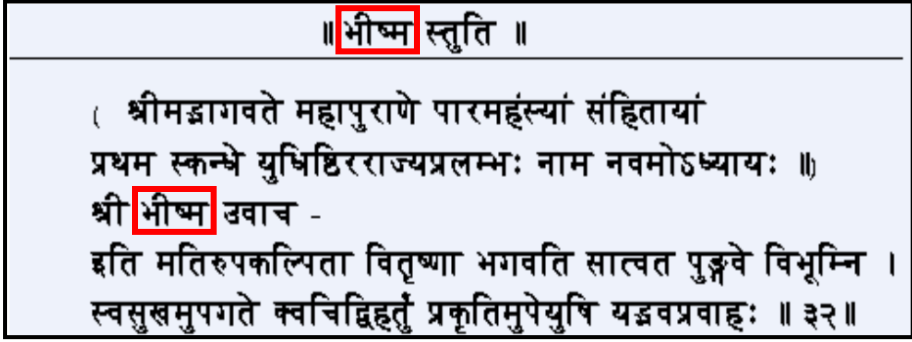
Keywords: Document Analysis, Keyword Spotting, Optical Character Recognition, Document Retrieval, Indic Scripts.

1 Introduction

For decades, Optical Character Recognition (OCR) has been considered as the primary enabling technology for automatic interpretation of handwritten or machine print document images. Given the relatively easy access to large numbers of document images in English and certain oriental scripts, OCR solutions have primarily focused on these scripts. However, the information boom of the last decade has led to a remarkable growth in the digital collection of documents in non-European and non-oriental scripts such as Indic scripts. Unfortunately, the progress in recognition technologies for Indic scripts such as Devanagari (used for Sanskrit, Hindi and other languages) has not been at par with the growth in the digital document collections. This can be attributed to a number of challenges in the form of poor quality documents, complex nature of the scripts, and relatively fewer years of research on Indic OCR.

Fortunately, recent advances in information retrieval have led to the emergence of Keyword Spotting as a viable method and an alternative to full OCR. Keyword Spotting essentially finds all occurrences of a typed input word in a set of handwritten/printed documents. Using this method, it is possible to obtain information from documents without relying on robust recognition strategies. Fig. 1 illustrates the concept where the boxed word image represents the spotted keyword.

G. Huet, A. Kulkarni, and P. Scharf (Eds.): Sanskrit CL 2007/2008, LNAI 5402, pp. 403–416, 2009.


॥ भीष्म स्तुति ॥

(श्रीमद्भागवते महापुराणे पारमहंस्यां संहितायां
प्रथम स्कन्धे युधिष्ठिरराज्यप्रलम्भः नाम नवमोऽध्यायः ॥)
श्री भीष्म उवाच -
इति मतिरुपकल्पिता वितृष्णा भगवति सात्वत पुङ्गवे विभूम्नि ।
स्वसुखमुपगते क्वचिद्विहर्तुं प्रकृतिमुपेयुषि यद्भवप्रवाहः ॥ ३२॥

Fig. 1. A sample Sanskrit document image with keyword boxed in red

We present two different approaches to Sanskrit Keyword Spotting. In the first approach, we describe a Block Adjacency Graph (BAG) based scheme for word recognition. It includes a BAG-based document clean up technique that uses a graph to maintain the overall ligature structure while removing noise that does not conform to a ligature shape. We use multiple hypotheses generated in the recognition phase to determine the similarity between a query word and a document word image. This ensures that even if the top choice result of an OCR is not correct, based on the similarity with the query, multiple word hypotheses returned by the OCR are considered. The second approach extends the idea of Keyword Spotting to multilingual documents. We use a moment-based word matching technique which maintains a script invariant representation of all word images. Word matching is performed using the cosine similarity. We also employ a relevance feedback technique to refine the word spotting results. The rest of the chapter is organized as follows: Section 2 describes background work in this area. Section 3 explains the proposed methodologies. Section 4 discusses the scope of the proposed methods. Conclusions are drawn in Section 5.

2 Related Work

Existing Keyword Spotting approaches can be broadly classified into two categories (Fig. 2): (i) OCR based approaches and (ii) Image feature based approaches. The OCR based techniques are suitable only for applications where the documents are of good quality and involve relatively small lexicons. Thus, instead of indexing the OCR'ed text of images for keyword retrieval, approaches based on indexing intermediate OCR likelihood or matching distance have been proposed for retrieving English documents [1, 2, 3]. However, such OCR systems are still not available for Sanskrit. This is because of several challenges in Sanskrit word recognition including: (i) a large number of ligatures resulting from all possible permutations of graphemes, (ii) ligature shapes made of complex primitives that cannot be easily segmented using conventional approaches. (iii) multiple font styles, and (iv) poor quality documents. To the best of our knowledge, our method is the first that uses a recognition-based approach for Keyword Spotting in Sanskrit documents.

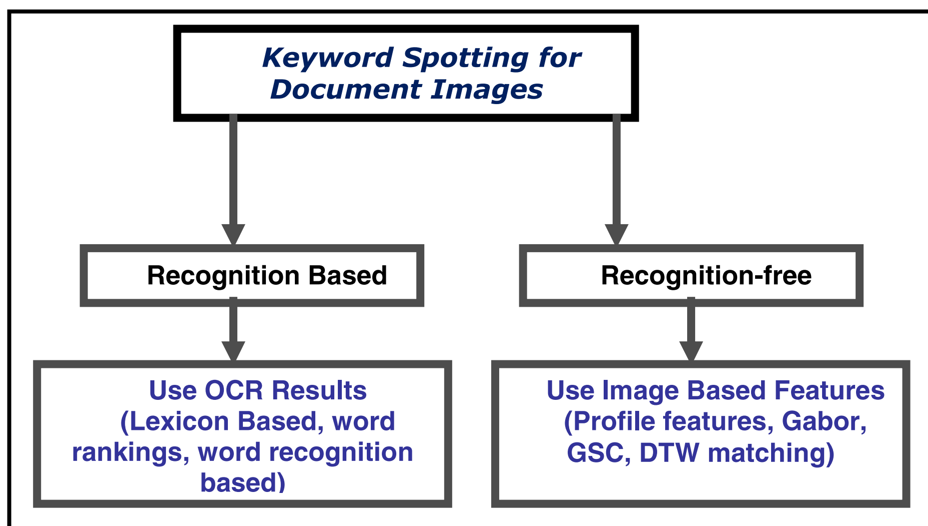


Fig. 2. Classification of existing Keyword Spotting techniques

In recognition-free Keyword Spotting approaches (Fig. 2), after preprocessing of document images and word segmentation, feature vectors are extracted from word images and stored in a database. When a user provides a query word, the similarity between the query and the word image in the database is computed, and word images are returned in decreasing order of similarity [4, 5, 6]. The comparison functions accommodate inexact matches by using Dynamic Time Warping (DTW) or string edit distance measures. Some methods use probabilistic similarity metrics or clustering to compute the similarity between the query word image and the document word image [6, 7]. Global shape features have been found to be not as discriminative as using an OCR specific to the script, and are prone to failure in the case of scripts like Devanagari that have a large and complex set of ligatures. The extracted global word shape features are not very reliable in poor quality Sanskrit documents.

Manmatha et al. [8] study various types of features for indexing handwritten English documents including profile-based features, gradient features and Gabor-based features [7]. Dynamic Time Warping (DTW) is used for finding similarity between the query word image and other word images. Though these features work fairly well with English documents, they have proved to be inadequate for Sanskrit given the complexity of class labels calling for more specific modeling of word images. Moreover, presence of the top horizontal line (head line) or "Shirorekha" renders most of the profile-based features ineffective. Also, DTW based approaches are slow. The Keyword Spotting method for Sanskrit proposed by Harish et al. [9] uses a Gradient, Structural and Concavity (GSC) feature set to measure the image characteristics at local, intermediate and large scales. The method specifies a sliding window approach. Local image features are computed from every window. However, in the presence of noisy word images, feature extraction is fragile. Template-based approaches such as these perform image matching in the feature space and hence require an image based model of the query word (template). Therefore the method does not scale with a large number of query words.

3 Proposed Methodologies

This section describes two different approaches to Sanskrit Keyword Spotting. The first is based on a common framework for both recognition and retrieval of Sanskrit documents. A font-independent OCR generates results at the (sub) grapheme level (atomic graphs) for a given word image. These results are used by a document retrieval system which defines a 'string edit' based cost metric to align OCR results with an input query word. The second approach uses a script-independent representation of word images to match the input query word with every indexed word image and retrieves the corresponding documents. The latter approach is recognition free but needs a query template consisting of word images of each query word to be searched. The following subsections describe these methodologies in detail.

3.1 Recognition-Based Keyword Spotting

Our system (Fig. 3) consists of 3 phases: (a) Preprocessing phase, (b) Recognition phase and (c) Matching phase. In the preprocessing phase, the horizontal profiles of the document image are used to segment the document into line images. The vertical profile of each line image is used to extract individual word images. Since the document images are often of poor print quality, the extracted word images are noisy. Therefore, the word images are cleaned by performing a Block Adjacency Graph (BAG) based smoothing procedure before initiating the recognition phase [10].

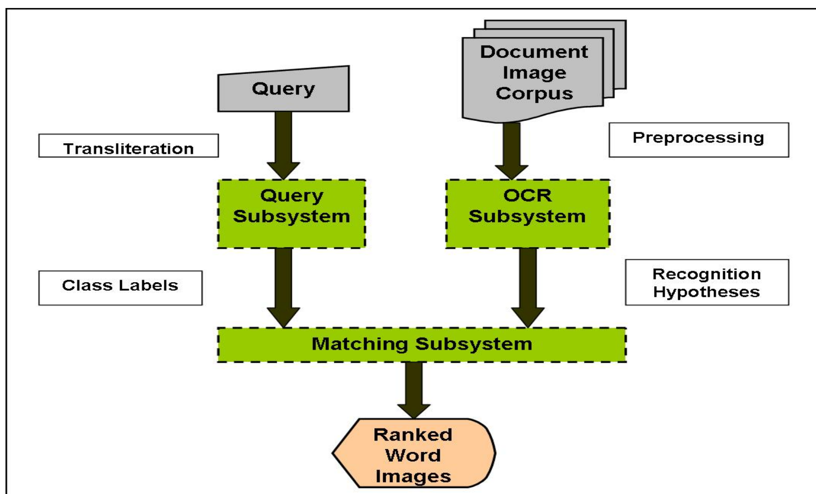


Fig. 3. System architecture for OCR based Keyword Spotting [10]

A Block Adjacency Graph (BAG) can be created by classifying runs as *merging*, *splitting* or *continuing* runs. BAGs can be constructed using both horizontal runs as well as vertical runs. Horizontal runs are classified based on the number of runs present above and below a given run. Runs that have one or no runs above them are labeled *splitting* runs, and runs that have one or no run below are labeled *merging* runs.

The remaining runs are classified as *continuing*. Adjacent runs which are classified as *continuing* are merged into a single block. A *splitting* run is added to the block above it, and a *merging* run is added to the block below it. A graph can be constructed by connecting the centroid of the neighboring blocks. Information pertaining to the image structure can be inferred from the edges in the graph. An example of BAG construction is shown in Fig. 4.



Fig. 4. BAG construction using horizontal runs for a sample grapheme [10]

The poor quality of many Sanskrit documents poses a significant challenge to word spotting. Noisy pixels on the contour boundaries and holes in the interior make feature extraction inaccurate. Image enhancement techniques like morphological operations or filters often cannot discriminate between the noise and natural ligature structures like junctions or holes. We perform BAG-based contour smoothing [10] which takes into account the junctions and holes that are inherent in ligature shapes. In this method, a pixel is modified not only by using structuring elements, but also by considering its connectivity to other parts of the ligature. Each pixel of an image can be mapped to a block in the BAG structure. If the corresponding BAG has multiple edges, it indicates that the block is connected to many other blocks of the graph. This implies that the pixel is likely to be a part of a joint that connects two or more parts of the image. In our implementation, we compute the average stroke width of the horizontal runs and the average block size. Blocks which have an area smaller than the average block size and are connected by only one edge are flagged as noise and removed. Since merging and splitting runs are grouped into larger adjacent blocks, thin sections at the trailing ends of a ligature are retained. Fig 5 shows an example of how noisy pixels are removed using this method.

In the recognition phase, the cleaned documents are input to a font-independent Devanagari OCR [10]. The OCR generates multiple segmentation hypotheses by selecting different combinations of blocks from the BAG. A naive approach is to exhaustively segment sub-graphs from the BAG. Such a scheme is computationally inefficient, since the number of possible hypotheses grows exponentially. The hypotheses search space is pruned by using the following constraints: (i) The primitives in the class space are assumed to be joined from left to right or top to bottom, and (ii) all of the primitives are assumed to consist of a single connected component. The equivalent conditions for sub-graph selection are: (i) A node cannot be selected if it is to the right (bottom for descender ligatures) of a node that has not been selected; and (ii) sub-graphs representing primitives in our class space cannot have disconnected nodes.

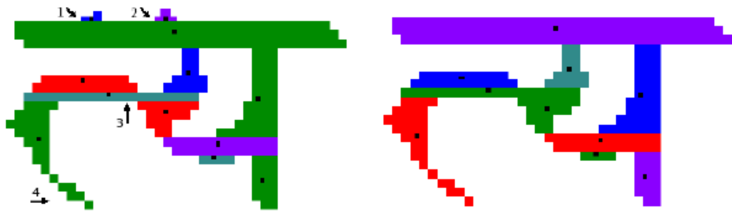


Fig. 5. Illustration of BAG-based noise removal – Left: (1, 2) sample with noise (3) thin joint (4) thin ligature boundary, Right: result of BAG-based noise removal [10]

A neural network based classifier is used to output recognition scores corresponding to every segmentation hypothesis. Fig 6 (Left) shows the Devanagari primitives classified by our OCR. Fig. 6 (Right) shows a sample segmentation hypotheses lattice generated by the sub-graph selection scheme.

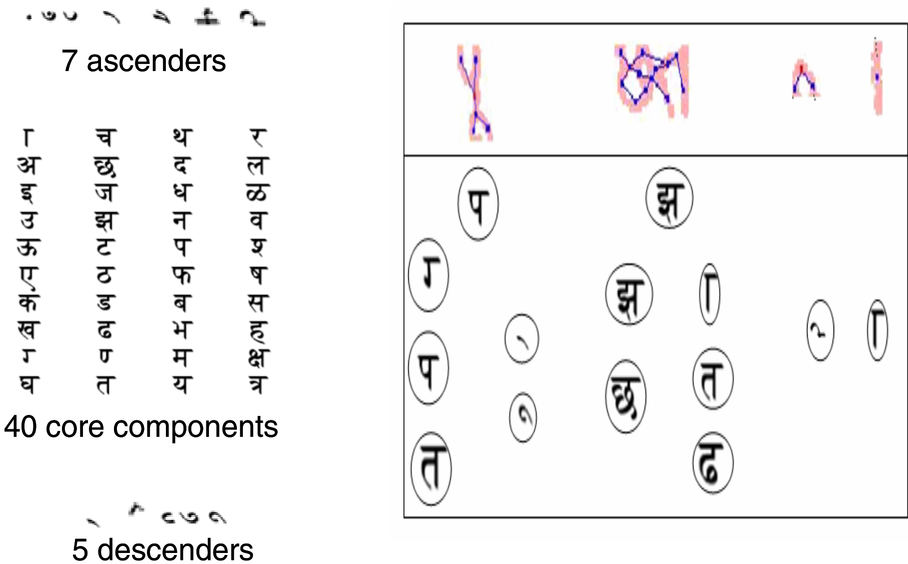


Fig. 6. Left: Devanagari OCR class space. Right: Segmentation hypotheses lattice generation using sub-graph selection.

We make use of the multiple recognition hypotheses provided by the OCR for each word in the document image. A similarity metric is designed to obtain the best match between the hypotheses and class labels corresponding to the query word. Therefore, the task of Keyword Spotting involves finding the best match between the input query word and the hypotheses. All the occurrences of the query word in the document are found by ranking the candidate words using a cost-based model. This is based on 'string edit' distance and represents the cost of aligning a combination of recognition hypotheses for a given word image with the query word.

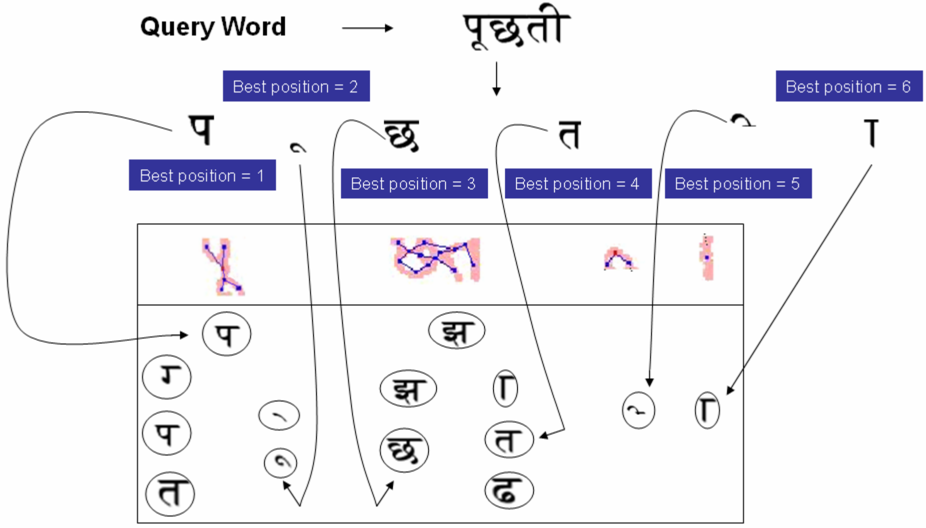


Fig. 7. Hypothesis search for best position of class labels

Since the hypotheses are associated with classifier scores for class labels for the Sanskrit graphemes at each position, our model incorporates both (i) classifier score and (ii) class label position within a hypotheses combination. A class label "*c*" here represents the Unicode value of a single Sanskrit grapheme present in the query text. The model takes a sequence of the desired class labels (ordering of individual Sanskrit graphemes in the query word), generated by transliteration of the queried text. For each class label in this sequence, the model finds the best position of occurrence of the class label among all the hypotheses corresponding to a single word image of the document (Fig. 7). It marks this position as the "*BestPos*" found and computes the difference "*MinDiff*" between the desired position and the actual position of the class label. It also stores the classifier score "*CScore*" which is the OCR output for this position of the class label. Thus the model computes the alignment cost. All the words are ranked in order of increasing cost and constitute the Keyword Spotting output. The overall cost model is as shown below:

$$MinDiff(c) = BestPos(c) - \mathrm{Re}\,qdPos(c)$$

$$\alpha(c) = \frac{1}{1 + MinDiff(c)}$$

$$Cost(w \mid q) = \sum_c 1 - (\alpha(c) * CScore(c))$$

3.2 Recognition-Free Keyword Spotting

The recognition-based approach to Keyword Spotting is very promising but has the disadvantage of being specific to a single script type. Digital libraries contain many

documents that involve multiple scripts. Our goal is to enable access to these documents without having to use a script specific solution for every script in the document collection. A script invariant representation of document images is needed to achieve this goal. Fig. 8 illustrates such a process where the input query gets translated into multiple languages and a script independent Keyword Spotting solution searches the entire collection for multiple translations and returns the relevant documents from each script.

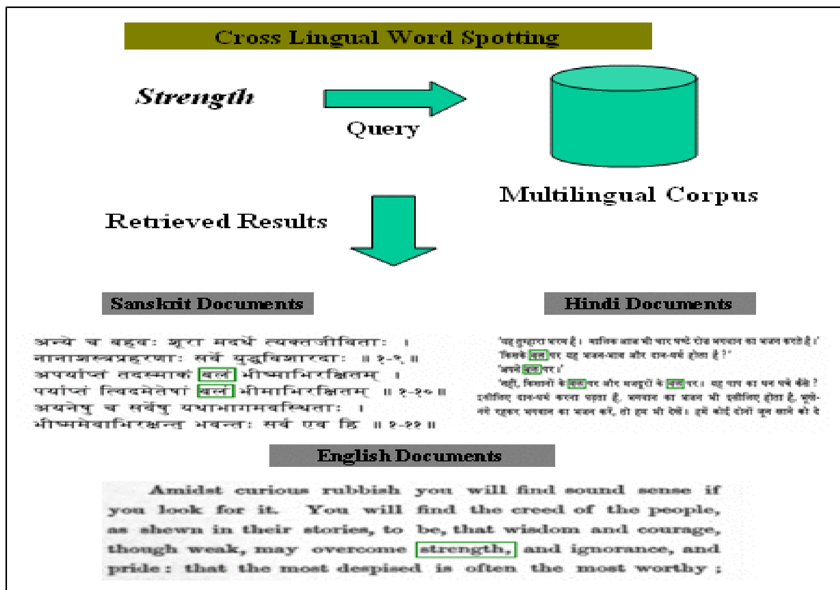


Fig. 8. Script independent Keyword Spotting solution

We present a script independent solution using image based features which are invariant to script type, image scale and translation. Previously proposed structural features such as GSC are not suitable as they are dependent on scale. Filter based features such as Gabor features have similar issues. Moreover, filter based feature extraction schemes are computationally expensive. We propose the use of Moment-based image features which have been previously used to achieve an invariant representation of a two-dimensional image pattern [11]. Geometric moments also have the desirable property of being invariant under image translation, scale and stretching and shear in either X or Y direction. Mathematically, such affine transformations are of the form of $X^* = aX + b$, and $Y^* = cY + d$ [12].

Geometrical Moments (GM) of order $(p+q)$ for a continuous image function $f(x,y)$ are defined as:

$$M_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x,y)\, dx\, dy$$

The function f(x,y) has only two possible values of 0 and 1, thus

$$M_{pq} = \sum_X \sum_Y x^p y^q f(x, y)$$

The center of gravity of image has co-ordinates:

$$\bar{x} = \frac{M_{10}}{M_{00}}, \bar{y} = \frac{M_{01}}{M_{00}},$$

If we refer to the center of gravity as origin, we get:

$$\bar{M}_{pq} = \sum_X \sum_Y (x - \bar{x})^p (y - \bar{y})^q f(x, y)$$

These moments are also referred to as Central Moments and can be expressed as combinations of moments of lower order. The variances of the moments are defined as:

$$\sigma_x = \sqrt{\frac{\bar{M}_{20}}{M_{00}}}, \sigma_y = \sqrt{\frac{\bar{M}_{02}}{M_{00}}},$$

They are used to normalize the co-ordinates by setting:

$$x^* = \frac{(x - \bar{x})}{\sigma_x}, y^* = \frac{(y - \bar{y})}{\sigma_y},$$

Using the normalized values of coordinates as obtained in equation above, the moment equation is as follows:

$$m_{pq} = \frac{\sum_X \sum_Y (x^*)^p (y^*)^q f(x, y)}{M_{00}}$$

This moment equation is invariant under image translations and scale transformations. Using this equation, we extract moment features up to the 7^{th} order. A feature vector consisting of 30 moment values corresponding to each word image is stored in the primary index. The index construction is followed by query template construction. A set of query words is chosen and a template is created which maps each query word to a query feature vector. A similar feature extraction scheme is employed to generate query feature vector from the query word image.

A standard Vector Space Model is used to represent the query word and all the candidate words. The index is maintained in the form of a word-feature matrix, where each word image "*w*" occupies one row of the matrix and all columns in a single row correspond to the moment values computed for the given word image. When the user enters any query word, a lookup operation is performed in the stored template to obtain the corresponding normalized word image for the input text. Feature extraction is performed on the word image to construct the query feature vector "*q*". A cosine similarity score is computed for this query feature vector and all the rows of the word-feature matrix.

The cosine similarity is calculated as follows:

$$SIM(q, w) = \frac{\vec{q} \cdot \vec{w}}{|\vec{q}| * |\vec{w}|}$$

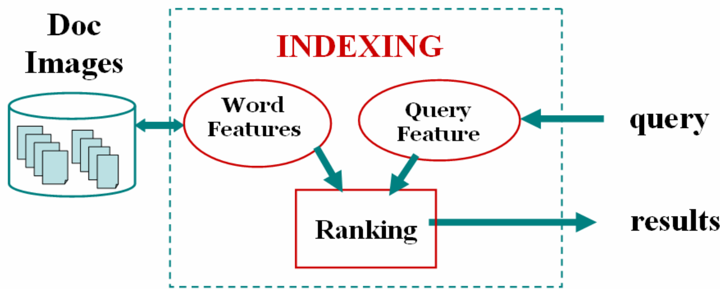


Fig. 9. Similarity matching using Cosine similarity

Fig. 9 describes the overall procedure for similarity matching. Since the word images present in the document corpus can be of poor print quality and noisy, the moment features computed may not be effective in assigning a high rank to all relevant word images. Also, the presence of higher order moments may be unstable. To overcome this limitation, we also implement a Relevance Feedback mechanism based on Rocchio's formula. This mechanism re-formulates the query feature vector by adjusting the values of the individual moment orders present in the query vector. The relevance feedback mechanism assumes a user input after the presentation of the initial results. A user enters a 1 for a relevant result or a 0 for an irrelevant result. The new query vector is computed as follows:

$$q_{new} = \gamma . q_{old} + \frac{\alpha}{|R|} . \sum_{i=1}^{i=R} d_i - \frac{\beta}{|NR|} . \sum_{j=1}^{j=NR} d_j$$

In the above equation, α, β and γ represent term re-weighting constants and *R* and *NR* represent relevant and non-relevant word images respectively. Fig. 10 describes the relevance feedback mechanism.

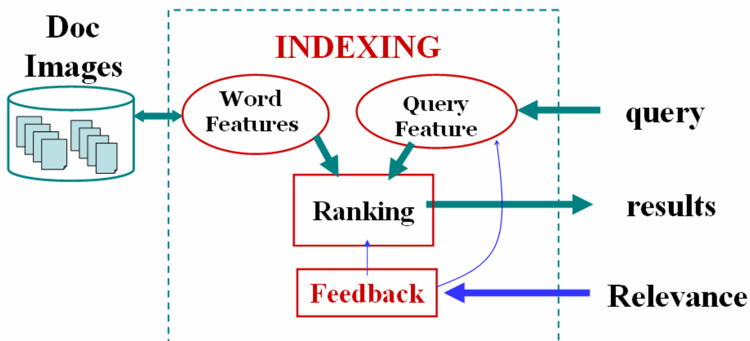


Fig. 10. Relevance Feedback using Rocchio's formula

Fig. 11. Devanagari Keyword Spotting results for 2 queries (OCR based approach)

4 Discussion

The recognition-based strategy is lexicon-free because the segmentation hypotheses are generated at the ligature-primitive level. We do not assume any query specific template and therefore our method is extensible to multiple queries. To evaluate this method, we implement some of the previously reported template-based word spotting methods. Three feature extraction schemes are studied:

– **Gabor Features:** Arrays of Gabor filters are designed corresponding to different scales and orientations. We experimented with 2 scales and 8 orientations [0, 45 90, 135, 180, –45, –90, –135], making a total of 16 Gabor filters. To extract features from every image, we convolve the input image with every Gabor filter and generate an output image. A block-based feature extraction mechanism is performed by dividing the output image into 5 vertical zones. For every vertical zone, we find the centroid of the image block and compute a weighted Gaussian based positive and negative component of the filter output. This yields 2 features per image block per filter, resulting in a total of 5*2 = 10 features per filter. By

applying all the filters in a similar fashion, we obtain a feature set of 160 features per image.

- **Gradient Features:** Images are divided in a 3×3 region and the gradient magnitude and direction is computed in each block using a Sobel operator. A gradient map is computed by applying an empirical threshold on the gradient magnitudes. The gradient directions in each cell are quantized into 8 bins: horizontal, vertical, and the two diagonals in each direction. Concatenation of the gradient vectors for the 9 cells forms a 72-element feature vector.
- **Gradient, Structural and Concavity Features:** As reported in [9], GSC has 512 features that represent three types of image attributes: (a) 192 gradient features, which represent the local edge and curvature information of the image, (b) 192 structural features, representing short strokes of different angles, and (c) 128 concavity features which reflect holes and concavities that are oriented in different directions.

प्रारब्धं भुज्यमानो हि गीताभ्यासरतः सदा ।
स मुक्तः स सुखी लोके कर्मणा नोपलिप्यते ॥२॥
महापापादिपापानि गीताध्यानं करोति चेत् ।
क्वचित्स्पर्शं न कुर्वन्ति नलिनीदलमम्बुवत् ॥३॥
गीतायाः पुस्तकं यत्र यत्र पाठः प्रवर्तते ।
तत्र सर्वाणि तीर्थाणि प्रयागादीनि तत्र वै ॥४॥
सर्वे देवाश्च ऋषयो योगिनः पन्नगाश्च ये ।
गोपाला गोपिका वापि नारदोद्धवपार्षदैः ॥
सहायो जायते शीघ्रं यत्र गीता प्रवर्तते ५॥
यत्र गीताविचारश्च पठनं पाठनं श्रुतम् ।
तत्राहं निश्चितं पृथ्वि निवसामि सदैव हि ॥६॥
गीताश्रयेऽहं तिष्ठामि गीता मे चोत्तमं गृहम् ।
गीताज्ञानमुपाश्रित्य त्रींलोकान्पालयाम्यहम् ॥७॥
गीता मे परमा विद्या ब्रह्मरूपा न संशयः ।
अर्धमात्राक्षरा नित्या स्वानिर्वाच्यपदात्मिका ॥८॥

Fig. 12. Sanskrit Keyword Spotting results for a query (Script-independent approach)

Two templates are designed to test the performance of template-based matching. The first template "A" is designed using fonts that are different from fonts seen in the document image collection. The second template "B" is constructed by selecting word images from the document collection itself. Experimental results show that in both cases, the recognition-based approach outperforms the template-based methods. Moreover, we observe a large variation in the performance of the template-based methods. Template B shows better performance as compared to template A, since the query word images in template A have the same font as the document image collection. Fig 11 shows results for two sample queries obtained using the OCR-based approach described above. However, this is a script specific solution which does not generalize over scripts other than Devanagari. An error analysis of this method suggests that most of the errors are due to the misclassification of ligatures by the

OCR as a result of confusion between classes that are similar in appearance. For instance, in the case of keywords such as टोलि (Toli) and सोने (Sone), [illegible] is often incorrectly recognized as [illegible]. Such confusion is also observed between र and स. One possible solution to this problem is to use a confusion matrix for post-processing the OCR results and to integrate this with the retrieval mechanism.

A script independent solution is desirable when dealing with multi-script document collections. The solution described in section 3.2 is recognition-free but requires a query template for every query into the corpus and is not scalable to multiple queries. Moreover, this method suffers from the same drawback of font-dependence as mentioned above. Fig. 12 shows the result for a single query on a Sanskrit document using the script-independent approach.

5 Conclusion

We have described two different approaches for Keyword Spotting in printed Sanskrit documents. The first approach which is script-specific uses Devanagari OCR based on Block Adjacency Graph (BAG) representation of word images. The recognition phase selectively segments ligatures by extracting sub-graphs from the initial BAG representation of the input word image. Multiple segmentation hypotheses are retained in the form of grapheme class labels and scores returned by the OCR. Finally, a string edit distance based cost model is used to retrieve the relevant documents by computing the alignment cost between the query word and multiple segmentation hypotheses for candidate words in the document corpus. The second approach, which is script-independent, uses an image moment based Keyword Spotting method to handle multi-script document collections.

Acknowledgment. This material is based upon work supported by the National Science Foundation under Grant No: IIS-0112059, IIS-0535038, and IIS-0849511.

References

[1] Howe, N.R., Rath, T.M., Manmatha, R.: Boosted decision trees for word recognition in handwritten document retrievals. In: Proceedings of the SIGIR, pp. 377–383 (2005)

[2] Lee, D.R., Kim, W.Y., Oh, I.S.: Hangul document image retrieval system using rank-based recognition. In: Proceedings of the International Conference on Document Analysis and Recognition, vol. 2, pp. 615–619 (2005)

[3] Rath, T.M., Manmatha, R., Layrenko, V.: A search engine for historical manuscripts. In: Proceedings of the 27th annual international ACM SIGIR conference on research and development in information retrieval (2004)

[4] Burl, M., Perona, P.: Using hierarchical shape models to spot keywords in cursive handwriting. In: IEEECS Conference on Computer Vision and Pattern Recognition, pp. 535–540 (1998)

[5] Decurtins, J.L., Chen, E.C.: Keyword spotting via word shape recognition. In: Vincent, L.M., Baird, H.S. (eds.) Proceedings of SPIE Document Recognition II, vol. 2422, pp. 270–277 (1995)

[6] Rath, T.M., Manmatha, R.: Word image matching using dynamic time warping. In: Proceedings of the Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 521–527 (2003)
[7] Cao, H., Govindaraju, V.: Template-Free Word Spotting in Low-Quality Manuscripts. In: Proceedings of the 6th International Conference on Advances in Pattern Recognition, pp. 135–139 (2007)
[8] Rath, T., Manmatha, R.: Features for Word Spotting in Historical Manuscripts. In: Proceedings of the 7th International Conference on Document Analysis and Recognition, pp. 218–222 (2003)
[9] Srihari, S.N., Srinivasan, H., Huang, C., Shetty, S.: Spotting Words in Latin, Devanagari and Arabic Scripts. Vivek: Indian Journal of Artificial Intelligence (2006)
[10] Bhardwaj, A., Kompalli, S., Setlur, S., Govindaraju, V.: An OCR based approach to word spotting in Devanagari documents. In: Proceedings of the 15th SPIE - Document Recognition and Retrieval, vol. 6815 (2008)
[11] Teh, C.-H., Chin, R.T.: On Image Analysis by the Methods of Moments. IEEE Trans. Pattern Analysis and Machine Intelligence 10(4), 496513 (1988)
[12] Alt, F.L.: Digital Pattern Recognition by Moments. The Journal of the ACM 9(2), 240–258 (1962)

The Phonemic Approach for Sanskrit Text

R.K. Joshi[1,*], T.N. Dharmadhikari[2], and Vijay Vasudev Bedekar[3]

[1] Centre for Development of Advanced Computing, (C-DAC)
Juhu, Mumbai, India
rkjoshi@cdacmumbai.in
[2] Vaidika, Sanshodhan
Mandal, Pune, India
[3] Institute for Oriental Study,
Thane, India

Abstract. Professor Joshi proposes that a phonemic encoding scheme be adopted as the standard for machine processing of Sanskrit text. In the scheme he details, each phoneme is represented by a single character code that represents a single Sanskrit sound. Graphic units in Devanagari corresponding to syllabic units, including consonant plus /a/, are represented as sequences. Glyphs corresponding to intial vowels verses dependent vowels are not given distinct encodings; rather they are selected based upon context. (P. Scharf)

1 Varṇa (Phoneme) and Akṣara (Syllable)

Sanskrit Grammar has distinguished the terms varṇa (phoneme) and akṣara (syllable). Both these terms are used in the context of spoken languages and can be ex-tended to written languages.

Since the oral tradition in India was of a higher order, the stress on right pronunciation was laid at most on the spoken language. To represent such speech nuances in written language, various cihna (signs) were evolved to strike the equivalence in spoken and written expressions. This extra-ordinary activity is part of the Indian tradition.

Therefore, the realization of such phonemic system in the context of new technology seems to be imperative where writing is talked in the context of speech and speech in the context of writing. The attempt is made to identify varṇamālā comprising of basic speech sound units as vowel phonemes (svara varṇa) and consonant phonemes (vyañjana varṇa) (Diagram 1).

These phonemes (varṇas) when combined as C...C + V or only V, form complete phonetic cluster. The correspondence in spoken and written syllables must be preserved through the Phonemic scheme firstly by giving each phoneme a distinct identity and secondly by giving each cihna –denoting nuances of speech –a distinct identity in the form of standardisation.

2 Sanskrit Phonology and Orthography

Presently, Devanagari script is used for writing classical Sanskrit as well as Vedic Sanskrit. This includes the multi-tier usage of diacritic marks of complex composi-tions, above, below and at the sides of the base glyphs. Therefore, as compared to modern historical derivatives from Sanskrit such as Hindi, Marathi, Nepali etc., the Sanskrit

[*] Deceased.

G. Huet, A. Kulkarni, and P. Scharf (Eds.): Sanskrit CL 2007/2008, LNAI 5402, pp. 417–424, 2009.

Diagram 1

Proposed Varnamala for computerization of Sanskrit

Vowels:

अ	आ	इ	ई	उ	ऊ	ॠ	ॠ	ऌ	ॡ
a 0061	ā 0101	i 0069	ī 012B	u 0075	ū 016B	ṛ 1E5B	ṝ 1E5D	ḷ 1E37	ḹ 1E39
ऎ	ए	ऍ		ऐ	ऒ o(Short)	ओ	ऑ o(Candra)		औ
e 0065	ē 0113	æ 00E6	ǣ 01E3	aj code needed	o 006F	ō 014D	œ	œ̄	au code needed

Vowel signs:

◌ं Anuswara	◌ँ Candra Bindu	◌ः Visarga
ṁ 1E41	m̐ 006D 0310	ḥ 1E25

Consonants:

क् K	ख् Kh	ग् G	घ् Gh	ङ् Ng	च् C	छ् Ch	ज् J	झ् Jh	ञ् Ny
k 006B	kɦ 006B 0266	ɡ 0261	ɡɦ 0261 0266	ŋ 014B	c 0063	cɦ 0063 0266	j 006A	jɦ 006A 0266	ɲ 0272
ट् Tt	ठ् Tth	ड् Dd	ढ् Ddh	ण् Nn	त् T	थ् Th	द् D	ध् Dh	न् N
ʈ 0288	ʈɦ 0288 0266	ɖ 0256	ɖɦ 0256 0266	ṇ 1E47	t 0074	tɦ 0074 0266	d 0064	dɦ 0064 0266	n 006E
प् P	फ् Ph	ब् B	भ् Bh	म् M	य् Y	र् R	ल् L	ळ् Ll	व् V
p 0070	pɦ 0070 0266	b 0062	bɦ 0062 0266	m 006D	y 0079	r 0072	l 006C	ḻ 006D	v 0076
श् Sh	ष् Ss	स् S	ह् H						
ʃ 0283	ʂ 0282	s 0073	h 0068						

text demands adequate range of characters as well as exhaustive rendering rules to achieve the advanced typographic quality in Classical as well as Vedic Sanskrit text.

3 Standardisation Principles

The effective smallest unit of the Sanskrit writing system can be the phoneme (varṇa). The range of phonemes (varṇamālā) consists of 'svara varṇa' (Vowel Phoneme) and 'vyañjana varṇa' (Consonant phoneme). While 'svara varṇa' is self-powered and it is not dependent on any other element, the 'vyañjana varṇa' however, needs an addition of 'svara varṇa' to compose its syllabic entity. While 'svara varṇa' (V) can be written down as syllables ('akṣara'), other syllables are the outcome of the combination of 'vyañjana varṇa' and 'Vowel varṇa'.

4 Phoneme (Varṇa) to Syllables (Akṣaras)

As mentioned earlier phonemes are divided into two types: vowel phonemes (svara varṇa) and consonant phonemes (vyañjana varṇa'). They together broadly constitute

the varṇamālā which has been referred as a varṇa-samāmnāya. The orthographic representation of these varṇas is done in a systematic way. The combination of consonant phoneme and a vowel phoneme produces a syllable (akṣara). A cluster of glyphs emerges as an outcome of this process.

For example,

/k/ + /a/ = /ka/ syllable which is written ... क्+अ=क

/p/ + /ā/ = /paa/ syllable which is written प्+आ=पा

Please note that corresponding to each svara phoneme there is an akṣara which is its syllabic form.

Vowel phoneme अ आ इ ई etc.
Vowel syllable अ आ इ ई etc.

This similarity has unfortunately caused the non differentiation between varṇamālā and akṣaramālā in the present times. Recently some grammar books in Indian languages are attempting to explain the difference between varṇa and akṣara.

In the written text when the combination of CV occurs, only then the svara varṇa is rendered into a svara mātrā sign. Otherwise the Vowel varṇa is written in the text as it is (as independent vowel). Therefore vowel mātrā is just the rendering form of a sound of a svara varṇa in CV. However in the context of V, VV, VC, VCCC…. The svara varṇa remains as is ओ, आई, अच्

5 Rendering of Akṣaras (Syllables)

k-phoneme + /a/ = k-akṣara क्+अ=क

The syllables formed by adding vowel phonemes /a/, /ā/, /i/, etc. to the consonant phoneme are written by creating akṣaras. All svara phonemes are added to one consonant phoneme one by one. This concept is called Bārākhaḍī (literally "group of 12 syllables") in Marathi.

Thus the concept of extended range of 'Bārākhaḍī' (18 syllables) is achieved in the following way.

K(d) + vv1 = K + A = KA क्+अ=क

K(d) + vv2 = K + AA = KAA क्+आ=का

K(d) + vv3 = K + I = KI क्+इ=कि

K(d) + vv4 = K + II = KII क्+ई=की

K(d) + vv5 = K + U = KU क्+उ=कु

K(d) + vv6 = K + UU = KUU क्+ऊ=कू

K(d) + vv7 = K + Vocalic R = K(Vocalic)R क्+ऋ=कृ

K(d) + vv8 = K + Vocalic RR= K(Vocalic)RR क्+ॠ=कॄ

K(d) + vv9 = K + Vocalic L = K(Vocalic)L क्+ऌ=कॢ

K(d) + vv10 = K + Vocalic L = K(Vocalic)LL क्+ॡ=कॣ

K(d) + vv11 = K + E = KE (Short) क्+ऎ= कॆ

K(d) +vv12 = K + EE = KE क्+ए=के

K(d) + vv13 = K + E = K(Candra}E क्+ऍ=कॅ

K(d) + vv14 = K + AE = KAI क्+ऐ=कै

K(d) + vv15 = K + O = KO (Short) क्+ऒ= कॊ

K(d) + vv16 = K + O = KO क्+ओ=को

K(d) + vv17 = K + O = K(Candra)O क्+ऑ=कॉ

K(d) + vv18 = K + AU = KAU क्+औ=कौ

The combination of two forms (C&V) into a syllable, at times creates a new integrated shape or retains partial identity of both the forms.

Syllables can also be formed by adding vowel phonemes to a sequence of more than one consonant phonemes. These syllables are called joḍakṣaras or saṁyuktākṣaras. For example:

k-phoneme + y-phoneme + ā-phoneme = kyā
क्+य्+आ=क्या
s-phoneme + t-phoneme + u-phoneme = stu
स्+त् +उ=स्तु

It is important to note that the invariant element in this process is the set of phonemes. The variation occurs in the shape of glyphs written in various Indian scripts. For example, the phoneme /k/ and /0/ will result in the glyph shape where graphic element is added in front in Devanagari where as in Bengali script, graphic shape will be added in front and prior to the base glyph. Therefore this model can be extended to most of the Indian languages which have phonetic base. To sum up the proposed scheme calls for code points for consonant phoneme k as compared to the existing Devanagari code which provides code points for the akṣara - glyph ka. The proposed scheme is of additive nature (k + a) as compared to subtractive model. This scheme would allow unambiguous representation of the entire repertoire of characters required for creating the exhaustive Devanagari script syllabic range along with its phonetic values.

Sample text output (*Diagram 2*)

IndiX

Output of Sanskrit and Vedic Sanskrit text through inputs of Sanskrit Varnamala

सीडैक
CDAC
Centre for Development of Advanced Computing
Gulmohar Cross Road No. 9
Juhu, Mumbai 49
Tel: +91-22-262 01606
Fax: +91-22-262 32195

Level A : Ordinary Reading

त्यमू षु वाजिनं देवजूतं सहोवानं तरुतारं रथानाम् ।
अरिष्टनेमिं पृतनाजमाशुं स्वस्तये तार्क्ष्यमिहा हुवेम ॥

RV. 10. 178.

- तार्क्ष्यसाम

Level B : Reading with High Stress and Low Stress

त्यमू षु वाजिनं देवजूतं सहोवानं तरुतारं रथानाम् ।
अरिष्टनेमिं पृतनाजमाशुं स्वस्तये तार्क्ष्यमिहा हुवेम ॥

RV. 10. 178.

- तार्क्ष्यसाम

Level C : Speaking (Chanting) with full tonal variations

ओ ऽ ६ म् ॥ त्यमूषु ॥ वाजि॥ ना ऽ २३४५ म् ॥ देवजूता ऽ २३४ म् ॥
सहोवानं ता ॥ स्ता ऽ ३ ॥ रँ रथानाम् ॥
आरिष्ट ऽ ना ऽ २३४ इमीम् ॥ पृतना ऽ ३४३ जमाशूम् ॥ स्वस्त ॥ याइ ॥
तार्क्ष्यमिहा ऽ ३४३ ॥ हू ऽ ३ वा ऽ ५ इ मा ऽ ६५६ ॥२॥

A part from "Naradiya Siksa"
by Usha R. Bhise,
Bhandarkar Oriental Research Institute,
Poona 411 004, INDIA

6 Considerations: Varnamala

6.1 Through the varṇamālā approach the IPA equivalence for Sanskrit text (as well as other Indian language text) can be established as one to one correspondence. And hence can be mapped very easily.

6.2 Through the varṇamālā -Phonemic approach lexical order and sorting operation in the areas of dictionary etc. can be done in the logical and more efficient way.

6.3 The Phonemic scheme will help grammatically in context to Samaas and Sandhi features in Sanskrit.

6.4 Under the phonemic scheme the keyboard input procedure will be simplified by reducing keys for vowel mātrās.

6.5 The range of Vedic Sanskrit accent/tonal marks when identified, can be positioned with vowel varṇas.

6.6 The essential Swaraadi Anusvar and Swaraadi Visarga can be added in the standardization.

6.7 If necessary, svara varṇa and vyañjana varṇa bhedaka cihna can be added in extended Devanagari varṇamālā.
6.8 Essential punctuation marks as used in the modern times can also be added if necessary.

7 Conclusion

7.1 The new scheme of Phonemes (vowels varṇas and consonants varṇas) as basic characters, is nearer to the linguistic model of Sanskrit and serves all the linguistic needs.
7.2 The text-processing operations like indexing and sorting which are very important for information storage and retrieval on computers can be performed efficiently.
7.3 Speech synthesis can be facilitated as the nuances of speech are preserved through phoneme standardization.
7.4 An absolute requirement for any script standardization is that it should facilitate a computer system to take any valid sequence of underlying character codes and algorithmically render the appropriate visual form from a given repertoire of surface glyphs. In the case of Phonemic scheme, the required character shaping rules are well-formulated, and therefore essential rendering engine can be built based on this concept.
7.5 (Alekh 1984 NCST, Vividha 1985 NCST, Vidura 1987 NCST). Text processing applications based on phonemic approach have been successfully implemented in Turnkey jobs and are in use with Sanskrit Institutions like Bharati Samskrt Vidya Niketanam, Lonavla and is considered favorably by other Sanskrit Institutions and scholars. Vedic Sanskrit text has been enabled on IndiX at C-DAC Mumbai using Phonemic approach (Diagram 2).

Acknowledgements

Technology Development for Indian Languages (TDIL), Govt. of India.
LC Group, C-DAC Mumbai (www.cdacmumbai.in), Vinod Kumar, Sr. Software Specialist, C-DAC Mumbai.
Jui Mhatre, TypeFont Designer, Language Computing Group, C-DAC Mumbai.
Supriya Kharkar, Assistant TypeFont Designer, Lan-guage Computing Group, C-DAC Mumbai.

References

Kelkar, A.R.: Transliteration of South Asian languages, a brief review and a proposal for a standard. Centre of advanced study in linguistics, Deccan College, Pune (1989)
Handbook of the International Phonetic Association. Cambridge University Press, Cambridge (1999)
Naravane, V.D.: Bharatiya Vyavahar Kosh. Triveni Sangam (1961)
Ladefoged, P.: Vowels and Consonants. Blackwell publishers, UK (2001)
More, C., Kulkarni, A.: Chetna Marathi Vyakaran Va Lekhan. Chetna Publications, Mumbai

Primer, P.: Department of official languages, Ministry of home affairs, Govt. of India
Joshi, R.K., Prague: A unified phonemic code based scheme for effective processing of Indian languages. 23rd Internationalization and Unicode (2003)
Joshi, R.K.: Vedic Code, a draft, Vish-wabharat No. 7 (October 2002)
Shama Sastri, R., Rangacharya, K.: The Taittiriya Pratishakhya, Motilal Banarasidas, Delhi (reprint, 1985)
Shanbaug, S., Rau, D., Joshi, R.K.: An intelligent multi-layered input scheme for phonetic scripts. In: ACM International conference proceeding series, Hawthorne, New York (2002)
Allen, W.S.: Phonetics in Ancient India. Oxford University Press, London (1961)

Supporting References – Note 1

अ इ उ ण् । ऋ ऌ क् । ए ओ ङ् । ऐ औ च् । ह य व र ट् ।
ल ण् । ञ म ङ ण न म् । झ भ ञ् । घ ढ ध ष् । ज ब ग ड द श् ।
ख फ छ ठ थ च ट त व् । क प य् । श ष स र् । ह ल् ।

इति सुत्राण्यणादिसंज्ञार्थानि । हकारादिष्वकार उच्चारणार्थः ।
१ हलन्त्यम् ॥ १.३.३ ॥
उपदेशेऽन्त्यं हल् इत् स्यात् । उपदेश आद्योच्चारणम् ।

These aphorisms are meant for the formation of 'aṇ' and such other technical designations. The vowel 'a' in the letters 'ha' etc. (in the aphorisms) is (appended) for (case of) pronunciation.

There are 42 phonemes enumerated in these 14 Śivasūtras, wherein there are 9 Vowels, 1 Mahāprāṇa, 4 Semi-vowels, 5 Nasals, 20 Consonants and 3 Fricatives. The addition of phoneme /a/ (a-kāra) in phonemes other than vowels such as 'Ha' etc. is only for the convenience of utterance and as such the seeming resemblance of these phonemes (such as 'Ha' etc.) to syllables can be explained.

Supporting References – Note 2

...तेषां विभागाः पञ्चधा स्मृतः
स्वरतः कालतः स्थानात् प्रयत्नानुप्रदानतः
इति वर्णविदः प्राहुर् निपुण तन्निबोधत

by tone | Length | Place of articulation | Process of articulation | and Secondary features

(अनुनासिककादि)-शिक्षाप्रकाश

9.10 पाणिनीय शिक्षा
Edited & translated by A. Weber

सव्यञ्जनः सानुस्वरः शुद्धोवापि
with consonants | with nasalization | or pure stand alone vowel

स्वरोऽक्षरम्
is syllable

A vowel with a consonant, or even by itself, forms a Syllable
ऋक् प्रतिशाख्य
with the commentary of Uvata- उवट

स्वयं राजते इति स्वरः ।
अन्वग् भवति व्यञ्जनम् ॥ -पतञ्जलि

The concept of vārṇa and its features such as Svara, Kāla, Sthāna and Prayatna etc. have been already identified in Pāṇinīya Śikṣā. The definition of akṣara (with consonant, with nasalization or a pure vowel can be called akṣara) is observed in Uvaṭa's commentary on the Ṛk Prātiśākhya and in Patañjali’s Mahābhāṣya.

Supporting References – Note 3

(त्) विवृतकण्ठोत्थित विवाराघोषाल्पप्राणाख्य बाह्यप्रयत्नविशिष्ट श्वासध्वनिजनित उत्तरदन्तमूलाधो-भागस्थान जिह्वाग्रकरण स्पृष्टप्रयत्नार्धमात्रिक परांग-भूत वायु देवताक ब्राह्मण जातिक तकार ॥

(ई) संवृत कण्टोत्थित संवाराख्यबाह्यप्रयत्न सहित नादध्वनिजनित तालुस्थानात्युपसंहृत कल्पौष्ठ-सहित जिह्वा मध्यकरण विवृतप्रयत्न द्विमात्रिक ॥

(प्रचयस्य) सूर्यदेवताक शूद्रजाति तमोगुणसहित मध्यमांगुलि मध्यरेखान्यासयोग्य उच्चकल्पतद्विलक्षण क्रौञ्चक्वणणतुल्य मध्यमस्वरहेतुभूत सर्वांगस्थानोत्पन्न प्रचयस्वरगुणक अग्निदेवताक ब्राह्मण जातिक ईकार द्विमात्रिक विरामाः ॥

जिह्वाग्रेण तवर्गे दन्तमूलेषु ॥ ३८ ॥

‘ तवर्गे ’ कार्ये ‘ जिह्वाग्रेण ’ वर्णान् ‘ दन्तमूलेषु ’ स्पर्शयेत् ॥३८

जिह्वाग्रेण यथावस्थितेन उत्तरदन्तमूलेष्वधोभागे स्पर्शयाति। शिक्षायां तु—तवर्गं दन्त्यं नकारं तु दन्तमूलीयं चोक्त्वा नकारस्यापि कचिद्दन्त्यत्वमुच्यते । तदस्माभिर्नाश्रयितव्यम् । प्रातिशाख्यविप्रतिषिद्धत्वात् । तस्मान्नकारोपि सर्वत्र दन्तमूलीयः ॥



[1] तज्जिह्वामम्.

In Vedalakṣaṇam various types of Varṇakramas are documented for Ghanapāṭhas. In the example above, the detailed phonetic profile of consonant phoneme /t/ and vowel phoneme /i/ is self explanatory and vivid. The terms such as takāra, ikāra are used to indicate the concept of varṇa. In the Uccāraṇakalpa – section of Taittirīya Prātiśākhya the formation of articulate sounds and their production is described in detailed way.

Author Index

Agrawal, Muktanand 219
Arora, Vipul 200

Bedekar, Vijay Vasudev 417
Behera, Laxmidhar 139, 200
Bhadra, Manji 219
Bhardwaj, Anurag 403
Bhattacharyya, Pushpak 328
Breuel, Thomas M. 391

Cardona, George 1
Csernel, Marc 358

Dharmadhikari, T.N. 417

Govindaraju, Venu 403
Goyal, Pawan 139, 200, 287

Hellwig, Oliver 266
Howe, Christopher J. 380
Huet, Gérard 162
Hyman, Malcolm D. 253

Jaddipal, V. 339
Jha, Girish Nath 219, 239
Joshi, Prasad P. 278
Joshi, R.K. 417

Kiparsky, Paul 33
Kulkarni, Amba 139
Kulkarni, Malhar 306, 328

Mani, Diwakar 219
Mishra, Anand 127
Mishra, Diwakar 219
Mishra, Sudhir K. 219, 239

Oguibénine, Boris 320

Patte, François 358
Phillips-Rodriguez, Wendy J. 380

Robinson, Peter 346

Scharf, Peter M. 95
Setlur, Srirangaraj 403
Sheeba, V. 339
Singh, Surjit K. 219
Sinha, R. Mahesh K. 287
Subash 219

Varakhedi, S. 339

Windram, Heather F. 380